ADA USER JOURNAL

Volume 46
Number 4
December 2025

Contents

Quarterly News Digest

Alejandro R. Mosteo

Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza; Instituto de Investigación en Ingeniería de Aragón, Spain; email: amosteo@unizar.es

Contents

[Messages without subject/newsgroups are replies from the same thread. Messages may have been edited for minor proofreading fixes. Quotations are trimmed where deemed too broad. Sender's signatures are omitted as a general rule. —arm]

Preface by the News Editor

Dear Reader,

The Ada User Society has published its first yearly bulletin [1], summarizing a year of activities. If you are not yet a member, this is a good occasion to learn what the Society is about and consider joining.

I am happy to report a particularly rich Ada-related Tools section in this issue, with a large number of new library releases, some even with SPARK verification. Do take a look!

Finally, the nowadays omnipresent topic of artificial intelligence has made it into the Digest, as practitioners share their experiences using libraries [2] and AI assistants for Ada coding [3]. The experiences are mixed but the discussion is timely.

Sincerely,
Alejandro R. Mosteo

[1] "Ada User Society Yearly (2025) Bulletin", in Ada-related Resources.

[2] "Anybody Tried OpenAI-API from Ada?", in Ada-related Tools.

[3] "Do You Use Artificial Intelligence for Ada Coding?", in Ada Practice.

Obituaries

Mike Woodger

From: John Barnes
Subject: Mike Woodger
Date: 15 Oct 2025 15:24 CET
To: arg@ada-auth.org

I have just heard from Brian Wichman that Mike Woodger has died at the age of 102.

Mike was involved in the early days of the definition of Ada.

I recall that an early draft of the ARM was produced at NPL (National Physical Laboratory) by a team of three people: Brian Wichman, Mike Woodger and myself. Jean Ichbiah referred to it as the London draft.

The first Rationale for Ada was that for the Green language and was written in 1978. The final version for Ada 83 was not written until 1986. It was written by Jean Ichbiah, Robert Firth, Mike Woodger and myself.

PS does anyone know where Robert Firth is?

From: Tucker Taft
Date: Thu, 29 May 2025 21:03:51 +0000

> I have just heard from Brian Wichman that Mike Woodger has died at the age of 102.

> Mike was involved in the early days of the definition of Ada.

Sad to hear! He was helpful during the Ada 9X effort, I remember.

> PS does anyone know where Robert Firth is?

Robert was at the Software Engineering Institute (CMU) in the late 90's. I haven't heard where he went afterward ...

Ada-related Events

FOSDEM 2026, Call for Participation

From: Fernando Oleo Blanco (Irvise)
Subject: [Ada devroom] FOSDEM 2026, call for participation
Date: Mon, 27 Oct 2025 17:39:24 +0000
Newsgroups: forum.ada-lang.io

The proposal to have an Ada Devroom for FOSDEM 2026 has *not* been accepted. We will look now into getting a stand, just like a few years ago. Though I don't have high hopes with this as in the last couple of years the competition has been great for the available spots.

Nevertheless, there are some other accepted DevRooms that could accept Ada related topics. A few of those examples are:

- Browser and web platform devroom: someone could showcase fully verified Ada/SPARK code running on the browser using WASM.
- Security Devroom: self-explanatory
- GCC (GNU Toolchain): self-explanatory
- LLVM: GNAT-LLVM frontend anybody??
- Embedded, Mobile and Automotive: someone could talk about the guide released by NVIDIA on how to get SIL-4 certified using Ada/SPARK and present the language.
- Software Performance: Ada has some unique properties when it comes to performance, maybe you would like to showcase those? (To be honest I have had an idea about this for soooo long…)
- Confidential Computing devroom
- Microkernel and Component-Based OS

Additionally, FOSDEM has indicated that they have given preference to rooms that were shared and not about a single topic, which does not bode well for Ada… They have also given more prominence to larger format talks, so this has also had an impact on the number of accepted DevRooms…

From: Dirk Craeynest (DirkCraeynest)
Date: Mon, 27 Oct 2025 21:33:15 +0000

Very unfortunate, but indeed, the competition for DevRooms was fierce.

The FOSDEM team wrote among others:

"Note we received far more proposals (137) than we can accommodate, so unfortunately we had to leave out a number of good proposals, including from some teams who had a well-organized devroom in one of the previous editions. Note that this only means we skip those rooms this year, and we will definitely schedule some of them again in a next edition."

So, we shouldn't despair for the future, with 12 well organized Ada DevRooms in previous editions.

They also mentioned:

"Compared to previous editions, we have chosen to avoid most devrooms that were about a single project."

Note: they did not say they avoided DevRooms "about a single topic" but "about a single project". I don't interpret that as not boding well for Ada, quite to the contrary, as the program of our DevRooms has never been about a single project, but the available time was always shared over various different projects (and even different topics as well one might say).

Anyway, no Ada DevRoom at FOSDEM 2026…

Let's hope we manage to get an Ada stand, though there as well there will be much competition for the available slots.

As Fer wrote there are several other DevRooms where Ada related topics would fit well. Please consider submitting there. And if your proposal is accepted, please keep us all informed, and we'll maintain a list of Ada-related presentations at FOSDEM 2026…

From: A.J. Ianozi (AJ-Ianozi)
Date: Tue, 28 Oct 2025 02:03:02 +0000

I noticed there were only three language specific devrooms this year: Go, Python, and Rust. The dozen+ that were available last year are gone too.

I wouldn't be surprised if they were trying to move away from specific language oriented rooms in the long term

From: Dirk Craeynest (DirkCraeynest)
Date: Tue, 11 Nov 2025 08:45:00 +0000

FYI, our proposal for an Ada-related stand was submitted before the deadline last Sunday evening. Fingers crossed…

From: Dirk Craeynest (DirkCraeynest)
Date: Mon, 17 Nov 2025 07:26:18 +0000

Bad news: our proposal for an Ada stand at FOSDEM 2026 has not been accepted…

The FOSDEM Stand Team wrote:

"We are sorry to tell you that we cannot accept your proposal "Learn & Make with Ada" as a FOSDEM stand. There were a lot of good proposals (212), and only a limited number of stands. You can still apply to do a talk in one of the accepted devrooms, lightning talk or a BOF. You can find more information at https://fosdem.org/submit or https://fosdem.org/"

As suggested in the 2nd paragraph above and stated in Fer's reminder yesterday, check out other devrooms and see if any Ada/SPARK can be presented there! [1]

The other two options to get Ada content at FOSDEM 2026 are [2]:

- The lightning talk is a very popular format, used at many conferences, where speakers have a mere 15 minutes at their disposal to showcase an open source project, an idea, or a concept thereof.
- BOF stands for Birds Of a Feather who, as the saying goes, flock together. FOSDEM has three meeting rooms that may be booked in 30 or 60 minute blocks for discussions. All the meetings are public so anyone who is interested can attend if there is enough space.

We are interested in your feedback, and especially your proposals!

[1] https://fosdem.org/2026/news/2025-10-31-devrooms-announced/

[2] https://fosdem.org/2026/schedule/

Google Summer of Code & Ada

From: Fernando Oleo Blanco (Irvise)
Subject: [gsoc & Ada] Working on Open Source Ada Projects and Getting Paid?
Date: Wed, 8 Oct 2025 18:23:23 +0000
Newsgroups: forum.ada-lang.io

I saw that GCC v16 is moving to stage 3 next month
https://www.phoronix.com/news/GCC-16-Stage-3-Next-Month
This means that the compiler won't allow new substantial functionality. The GNAT/Ada part of GCC is a bit different and we tend to have more leeway in it. Nonetheless, do you think the work on parallel can be pushed for comments to the GCC mailing list?

From: Tucker Taft (sttaft)
Date: Wed, 8 Oct 2025 18:51:38 +0000

We are getting close. We have been trying to stabilize a chunk of functionality, and fix up the commit messages and the test suite so we could submit a patch. I would anticipate a couple of more weeks. Even before then we will make something available for folks who feel competent to clone a repository on GitHub and give them a try. Here is a branch you could try now:

https://github.com/Ada-Rapporteur-Group/gcc-mirror/tree/devel/arg-proto/parallel-exp-par-1

From: Tucker Taft (sttaft)
Date: Tue, 4 Nov 2025 13:33:20 +0000

We now have a "released" branch which supports parallel "for" and parallel "do" that is available for "beta" testing:

https://github.com/Ada-Rapporteur-Group/gcc-mirror/tree/devel/arg-proto/ada2022-parallel-release

From: Fernando Oleo Blanco (Irvise)
Date: Tue, 4 Nov 2025 19:31:26 +0000

Congrats on the work done! I suppose it does not cover the parallel 'Reduce, does it?

Also, for people interested in trying it out without having to build GCC from scratch, you can use @Max's builds which use this parallel branch! [1]

Have these patches been sent for initial review and possible admission in GCC 16?

[1] https://github.com/reznikmm/GNAT-FSF-builds/releases/tag/16.0.0-20251102

Ada Monthly Meetup November 2025

From: Fernando Oleo Blanco (Irvise)
Subject: Ada Monthly Meetup November 2025
Date: Thu, 23 Oct 2025 21:17:32 +0000
Newsgroups: forum.ada-lang.io

I would like to announce the November (2025) Ada Monthly Meetup which will be taking place on the 8th of November at 15:00 UTC time (16:00 CET). Attention! As daylight savings will kick into action shortly, your local time may have changed! As always the meetup will take place over at Jitsi. The Meetup will also be livestreamed/recorded to Youtube.

Important: due some personal scheduling issues, this meetup will be taking place the second week of the month. Also, there will be no Meetup in December due to some holidays in Spain, January, due to Christmas and February due to FOSDEM. Therefore, the meetup after this one will most likely be in March.

If someone would like to propose a talk or a topic, feel free to do so! We currently have no proposals. Nevertheless, I hope to talk about FOSDEM news (if any) and an Ada compiler for DOS as reported by @joakim-strandberg.

Here are the connection details from previous posts:

The meetup will take place over at Jitsi, a conferencing software that runs on any modern browser. The link is Jitsi Meet [1] The room name is "AdaMonthlyMeetup" and in case it asks for a password, it will be set to "AdaRules".

I do not want to set up a password, but in case it is needed, it will be the one above without the quotes. The room name is generally not needed as the link should take you directly there, but I want to write it down just in case someone needs it.

P.S: feel free to repost this in other forums or chats, such as Reddit! >:D

[1] https://meet.jit.si/AdaMonthlyMeetup

From: Fernando Oleo Blanco (Irvise)
Date: Sat, 8 Nov 2025 16:33:31 +0000

Here are the minutes of the meeting for today's reunion!

- @joakim-strandberg presented the Meridian Open Ada compiler. A compiler for DOS which anybody can

download freely and give it a test or a full drive! It has a full RTS (actually two!) and you could try it out for this year's Advent of Code. Additionally, it also comes with its own, full blown IDE :D! Here are the relevant links:

- Wiki: Meridian Open Ada - EDM2 [1]
- Download page (requires free login): Download Meridian Ada Z 4.1.4 by Meridian Software Systems, Inc. | VETUSWARE.COM - the biggest free abandonware collection in the universe [2]
- Manual of the compiler: Meridian Ada 4.1 User Guide (DOS version) : Free Download, Borrow, and Streaming : Internet Archive [3]
- Validation report of the Green Hills Compiler which has the definition of bamp (the linker used by the Meridian compiler): https://apps.dtic.mil/sti/tr/pdf/ADA288573.pdf
- And there are still DOS game jams, you could even use this compiler for it! DOSember Game Jam - itch.io [4]
- Nvidia published their certification process for the highest quality software that one can build for the automotive sector [5] But additionally, some people who work on the safety-critical systems of NVIDIA's chips, gave a presentation in DEF CON 33 (this year) detailing their work on the RISC-V architecture and Ada/SPARK: https://www.youtube.com/watch?v=KhWtkZmOPn4 It is heavily implied in their presentation that they have shipped over 1 billion cores running Ada/SPARK!!!
- @abitofhelp dropped a few new and cool libraries and tools, such as Functional Error Handling v1.0.0: Result, Option, and Either, Try_To_Result [6] Go and check them out!
- AWS is going into maintenance mode and it is projected to be archived in the future: [7] If you would like to keep maintaining it, contact @AJ-Ianozi as he is looking into options about it!
- Similar news about PolyORB, it is also getting deprecated in the future: [8]
- The VSS library has now been split into two! Read more about it here: [9]
- Advent of Code (AoC) is coming and this year it is going to be quite a bit shorter! [10]
- The last major Ada 2022 feature that was still not implemented in GNAT, parallel, is starting to see the light of day! [11] Read more about it in the link
- If you would like to test it out, you can use @Max's precompiled GNAT binaries: [12]
- It would be great for people to use this new feature in the AoC to test it out and find bugs before it is hopefully upstreamed to GCC 16!
- The Alire project now also provides compiled prereleases of GCC 16 in case you would like to start playing around with them! [13]
- We did not get a devroom accepted for FOSDEM, but we are going to see if we get a stand!
- A reminder that the next Ada-Europe international Conference will be taking place in Sweden, and we hope to have another Ada Developers Workshop! https://www.ada-europe.org/conference2026/
- Wesley and Weston are working hard on the recordings of the Paris' Ada Developers Workshop !
- And we hope to release some more news about the creation of a fund to… well, fund, Ada activities and achievements soon!
- We are trying to form a Committee that will manage the funds, select ideas and distribute the grants. We need at least one more person! If you are interested in participating in the Committee, please, let me (@irvise) and @sttaft know!

Thank you to everybody who attended! If you couldn't, you can find the livestream/recording in https://www.youtube.com/watch?v=F93rgu4t9As

[1] https://www.edm2.com/index.php/Meridian_Open_Ada

[2] https://vetusware.com/download/Meridian%20Ada%20Z%204.1.4/?id=17519

[3] https://archive.org/details/MeridianAda41UserGuideDOS/1_Readme/

[4] https://itch.io/jam/dosember-game-jam

[5] https://forum.ada-lang.io/t/nvidia-publishes-spark-process-to-meet-iso-26262-requirements/2114/3

[6] https://forum.ada-lang.io/t/functional-error-handling-v1-0-0-result-t-e-option-t-and-either-l-r-try-to-result/3925

[7] https://forum.ada-lang.io/t/aws-end-of-life/3858

[8] https://forum.ada-lang.io/t/polyorb-end-of-life/3927/7

[9] https://forum.ada-lang.io/t/ann-vss-has-been-split-into-two-libraries-and-main-repo-is-now-archived/3885

[10] https://forum.ada-lang.io/t/plans-for-aoc-2025/3930

[11] https://forum.ada-lang.io/t/gsoc-ada-working-on-open-source-ada-projects-and-getting-paid/1892/9

[12] https://github.com/reznikmm/GNAT-FSF-builds/releases/tag/16.0.0-20251102

[13] https://github.com/alire-project/GNAT-FSF-builds/releases

Swiss Ada Event 2025 Presentations

From: Gautier de Montmollin (zertovitch)
Subject: Swiss Ada Event 2025 - Presentations
Date: Sun, 16 Nov 2025 11:51:14 +0000
Newsgroups: forum.ada-lang.io

Recently, there was an Ada event at the Observatory in Schaffhausen, by Ada Switzerland and for its members.

The slides are now publicly available on Ada Switzerland's web site:

ada-switzerland.ch [1]

After the Ada-related stuff, there was a visit of the telescopes

…and a projection at the digital planetarium!

[1] https://ada-switzerland.ch/#SwissAdaEvent

[2] https://forum.ada-lang.io/uploads/default/6d12df329d7900d71b1e4ab16360f435127ae50f

Ada-Europe Conference - 2nd Call for Contributions - AEiC 2026

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Ada-Europe Conference - 2nd Call for Contributions - AEiC 2026
Date: Thu, 20 Nov 2025 15:30:28 +0000
Newsgroups: comp.lang.ada

[CfP is included in the Forthcoming Events Section —arm]

[Charity] Advent of Ada/SPARK 2025

From: Fabien (Fabien.C)
Subject: [charity] Advent of Ada/SPARK 2025 submissions
Date: Thu, 27 Nov 2025 17:25:47 +0000
Newsgroups: forum.ada-lang.io

On Monday, AdaCore will announce a new edition of the Advent of Ada/SPARK charity event. For each person completing one of the Advent of Code challenges using the Ada programming language, AdaCore will donate $10 to the Ada Developers Academy [1], up to a total of $5,000. And for those willing to go an extra mile, AdaCore will donate $20 if the solution is implemented in SPARK with at least proof of absence of run-time errors (a.k.a. Silver level).

Since there are fewer puzzles to solve this time, we decided to add a little bonus. For everyone who completes the full 12

puzzles, we add an extra $100 to the donation.

[...]

Happy hacking!

[1] https://adadevelopersacademy.org/

From: Jean-Pierre Rosen (rosen)
Date: Sat, 29 Nov 2025 09:02:26 +0000

Note that this Ada Developers Academy is named after Lady Ada, but it does not teach or use Ada the language. Too bad!

Ada-related Resources

[Delta counts are interpolated from October 1st to January 1st. Tabulated raw data is available at https://bit.ly/auj-signals —arm]

Ada on Social Media

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada on Social Media
Date: 22 Mar 2026 23:21 CET
To: Ada User Journal readership

Ada groups on various social media:

- Reddit: 1_655 active members [1]
- LinkedIn: 3_733 (+29) members [2]
- Stack Overflow: 2_448 (+5) question [3]
- Ada-lang.io: 442 (+38) users [4]
- Gitter: 295 (+2) people [5]
- Discord: 257 (+60) users [6]
- Telegram: 236 (+10) users [7]
- Libera.Chat: 78 (+3) concurrent users [8]

[1] https://old.reddit.com/r/ada/

[2] https://www.linkedin.com/groups/114211/

[3] https://stackoverflow.com/questions/tagged/ada

[4] https://forum.ada-lang.io/u

[5] https://app.gitter.im/#/room/#ada-lang_Lobby:gitter.im

[6] https://discord.gg/fEmGe87c

[7] https://t.me/ada_lang

[8] https://netsplit.de/channels/details.php?room=%23ada&net=Libera.Chat

Repositories of Open Source Software

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Repositories of Open Source software
Date: 22 Mar 2026 23:21 CET
To: Ada User Journal readership

GitHub: >7_994* (+194) repositories [1]
>1_000* (=) developers [1]
Alire: 1_651 (+219) releases [2]
615 (+30) crates [3]
Rosetta Code: 1_069 (+18) examples [4]
43 (=) developers [5]
Sourceforge: 241 (=) projects [6]
Open Hub: 214 (=) projects [7]
Codelabs: 69 (+5) repositories [8]
Codeberg: 26* (new) repositories [9]

*Bitbucket has been discontinued due to new search limitations. The popularity-gaining Codeberg site has been newly added.

[1] https://github.com/search?q=language%3AAda&type=Users

[2] https://alire.ada.dev/crates.html

[3] `alr search --list --full`

[4] https://rosettacode.org/wiki/Category:Ada

[5] https://rosettacode.org/wiki/Category:Ada_User

[6] https://sourceforge.net/directory/language:ada/

[7] https://www.openhub.net/tags?names=ada

[8] https://git.codelabs.ch/?a=project_index

[9] https://codeberg.org/explore/repos?q=ada&topic=false&language=Ada&only_show_relevant=false

Language Popularity Rankings

From: Alejandro R. Mosteo <amosteo@unizar.es>
Subject: Ada in language popularity rankings
Date:22 Mar 2026 23:21 CET
To: Ada User Journal readership

[Positive ranking changes mean to go up in the ranking. —arm]

- PYPL Index: 11 (0) 2.45% (+0.28%) [2]
- TIOBE Index: 18 (=) 1.06% (=) [1]
- Stack Overflow Survey: 38 (+2) 1.14% (+0.24) [3]
- IEEE Spectrum (trending): 43 (=) Score: 0.0040 (+0.002) [4]
- IEEE Spectrum (general): 45 (+1) Score: 0.0029 (+0.0002) [4]
- IEEE Spectrum (jobs): 38 (+2) Score: 0.0019 (+0.0002) [4]
- Languish Trends 166 (-1) 0.01% (=) [5]

[1] https://www.tiobe.com/tiobe-index/

[2] http://pypl.github.io/PYPL.html

[3] https://survey.stackoverflow.co/2024/

[4] https://spectrum.ieee.org/top-programming-languages/

[5] https://tjpalmer.github.io/languish/

WG9 Seeks Editor for Technical Report

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: WG9 seeks Editor for "Guide for Use of Ada in High Integrity Systems" (TR 15942 technical report)
Date: Wed, 8 Oct 2025 16:35:35 +0000
Newsgroups: comp.lang.ada

[Posting for Tucker Taft: -- dc]

WG9 is looking for an editor who could help produce a new version of TR 15942, which is a Guide for the use of Ada in High Integrity Systems. The editor should have been involved in the safety-critical processes that the TR covers (see below and ISO/IEC TR 15942 [1]) and be able to identify the mapping between the various standards requirements and Ada features; the wider the roles they have played the better.

As a bare minimum:

* Being an accomplished current Ada programmer.
* Having as much experience as possible in some high-integrity domain.

Ideally:

* Deep knowledge of modern Ada, familiarity with new features in Ada 2022, as much hands-on practice as possible, a current developer.
* Experience with certification processes under standards such as DO-178C, IEC 61508, EN 50128, other recent standards that may be relevant.
* Experience with embedded development and Ada profiles (Ravenscar, Jorvik) and restriction pragmas.
* Familiarity with at least a couple of Ada compilers to differentiate compiler-specific issues from language issues in the build process.
* Understanding/experience with run-time resource usage and measurements (WCET, stack usage, memory leaks, ...).
* Previous involvement with ISO standards and documents, writing/using standards or, at least, technical documents.

The current version is based on Ada 95, so some significant updates may be required. The PDF for the current document can be acquired at ISO/IEC TR 15942 [1] for free.

If this interests you, please send an email to my last name at adacore.com. If you have questions or comments, feel free to post them here.

[1] https://www.iso.org/standard/29575.html

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Date: Mon, 13 Oct 2025 07:59:49 +0000

[Posting for Tucker Taft: -- dc]

I am happy to report we have found an editor for this document - Niklas Holsti from Finland. Thanks, Niklas, for volunteering for updating this valuable document.

Looking for Volunteers for an Ada Funding Committee

From: Fernando Oleo Blanco (Irvise)
Subject: Looking for Volunteers for an Ada Funding Committee
Date: Mon, 17 Nov 2025 19:23:07 +0000
Newsgroups: forum.ada-lang.io

As we have been hinting in the Ada Monthly Meetup, we are creating (a hopefully lasting) funding system for Ada projects and improvements to the wider ecosystem. This is being done within the Ada User Society (those that attended the GA will have more info). We are trying to form a committee that will be guiding those funds. Currently, only @sttaft and I are part of it but at least a third one would be needed (to keep things with some variety). We can have up-to five members. But why is this needed and what does the committee do?

The funds

So we would like to set up a small fund with which to reward and encourage the development of valuable and high quality and impact projects in the Ada ecosystem. It will also reward seasoned developers as well as to encourage new people to take on some tasks. The funding part is mostly taken care of. But then the question becomes… What is considered valuable and with what amount of funds should things be awarded? That is where the committee and the base requirements come into place.

The requirements

Not everything can be eligible for a reward/grant. Only open [source] and projects available to the community can be awarded, as we want to maximize the impact and gains of the results to the wider community. But there are no restrictions on the nature and objective of the project. However, as the funds are limited, we will be encouraging/considering projects that are of high impact and can greatly benefit the Ada ecosystem. Think of compiler improvements, bringing Ada support to growing targets or ecosystems, improving tools that are widely used, packaging and making things easier for everybody, major projects that showcase Ada's potential, etc. Obviously, only quality results/code would be accepted for a grant, as "it works on my machine" is not good enough.

The Committee

In short, the Committee would take decisions through votes on what projects are worthy of funding and how much money would that be. The committee has to listen to the community's needs and Ada's future. Currently, we are considering two options to consider potential projects/outcomes that would be rewarded:

- Individuals from the community could propose tasks and ideas and the Committee would vote on whether it wants to allow for a grant or fund for them.
- The committee could propose tasks to be done and reward those that achieve them, something in the vein of a competition. Also, the Committee could grant funds to results/code that people have already done that it feels are worth awarding.

Due to the above, the people that form part of the Committee need to:

- listen to the community,
- assess the impact of the results or potential results of tasks,
- be impartial to the nature of the projects and the people that carry them out,
- manage the constraints of the budget,
- have a strong will to the further improvement of Ada as a language and as an ecosystem,
- be available to vote (most likely via online meetings or email) on the topics at hand,
- seek new funds with which to continue the activity,
- be willing to help.

So, with that out of the way… Do you have any questions? Would you like to take part in the Committee? And more importantly, why would you like to be part of such a committee?

From: Fernando Oleo Blanco (Irvise)
Date: Fri, 5 Dec 2025 09:45:10 +0000

Two very knowledgeable people have volunteered. I have been quite busy with work/personal things so I have not had much time to coordinate this, but I hope to get all in shape before the end of the year

Advent of Code 2025 Bonus

From: Max Reznik (Max)
Subject: Advent of Code 2025 Bonus
Date: Mon, 1 Dec 2025 18:13:05 +0000
Newsgroups: forum.ada-lang.io

I'd like to offer a series of short videos about Ada 2022 as a reward for participating in Advent of Code 2025. To participate, you must:

- register on the leaderboard [1] 1708445-6a8f7730
- have a GitHub account with valid SSH keys (RSA or ed25519).

The files with the video links are encrypted for each user and are located in a separate repository along with the Ada decryption program. You just need to build and run it:

```
git clone https://github.com/reznikmm/dead-man-hand
cd dead-man-hand
alr build
./bin/dead_man_hand
```

As an alternative to the Ada program, there's also a Python script.

PS If you are AI allergic then don't bother. I used Gemini Text-To-Speech to make the video.

PPS If you want to watch the video without solving Advent of Code, subscribe to my Patreon [2].

[1] https://adventofcode.com/2025/leaderboard/private/view/1708445

[2] https://www.patreon.com/c/ada_re

Ada-related Tools

AWS End of Life

From: Kevin Chadwick (kevlar700)
Subject: AWS end of life
Date: Wed, 1 Oct 2025 13:32:19 +0000
Newsgroups: forum.ada-lang.io

> It's not nasty. It's working as designed. Nasty are the people who grab open source and keep the derived work to themselves. All my Ada code is GPL licensed and intentionally so.

There is truth to virality being nasty and it actually reduces the chances of having users who may submit patches. By virality I think he means that if I copy a procedure or section of code out of a GPL3 library perhaps for compatibility then unless I make it a standalone library then legally I'm meant to open source all my own code that I copied it into. Though I'm not sure if that would actually stand up in court. In any case the BSDs do not use GPL code because of this virality issue. At least OpenBSD doesn't and we're stuck on an old GCC in base until LLVM came along and now that's changed its license to Apache I think and I'm not sure on the situation now. Having OpenBSD devs testing the latest GCC on base and not just ports would surely have been a good thing.

From: Frédéric Loyer (F-Loyer)
Date: Tue, 28 Oct 2025 21:38:24 +0000

Yes, the idea is that statically linking a GPL code to another implies the whole result is GPL. The code owner could claim that because you don't respect the licence, the copy is forbidden. The idea of copyright laws is that the copy is forbidden unless the author has given the right to you. And here, the right is conditional. (Like with many software that can be copied (from disk to memory)

with some fancy conditions : no more than 10 users connected, no more than 24 CPU-cores, etc).

An interesting case is ZFS (an Oracle file system) which can be used on Linux… but must be shipped separately because ZFS is not GPL. An adaptation layer linked to Linux (then GPL) has been developed.

Note that the issue of GPL libraries is quite annoying, and a derived license, LGPL has been developed to permit linking such a library to a proprietary code (the virality is limited to the library modifications). But it is up to the library owner to choose between GPL and LGPL.

From: Fabien (Fabien.C)
Date: Thu, 30 Oct 2025 09:14:36 +0000

Please be careful and precise when discussing licenses. AWS is under GPL3 + GCC Runtime Library Exception, which is quite different from the "pure" GPL.

[License text removed. —arm]

GNAT FSF Weekly Snapshots

From: César Sagaert (csagaert)
Subject: GNAT FSF weekly snapshots
Date: Thu, 2 Oct 2025 09:08:09 +0000
Newsgroups: forum.ada-lang.io

This short message to announce that the GNAT-FSF-builds [1] repo will now publish weekly builds of GCC snapshots for the next major release (currently 16).

These builds have no guarantees of quality or stability, and we will minimally investigate build failures (for instance, the Windows build has failed for the first two builds because of some forgotten warning.)

The Releases tab on the repo will list these snapshots under the gnat-16.0.0-snapshot prerelease, and for convenience we also publish an Alire index automatically under the snapshots-index branch. Instructions on how to set it up can be found in the README

[1] https://github.com/alire-project/GNAT-FSF-builds/tree/snapshots-index

From: Max Reznik (Max)
Date: Thu, 2 Oct 2025 09:52:21 +0000

That's great!

BTW Could you share what's wrong with 15.2 for Mac OS X, arm64? I saw that there is no corresponding toolchain in the Alire.

From: César Sagaert (csagaert)
Date: Thu, 2 Oct 2025 09:57:11 +0000

We base the macOS ARM releases for GNAT 15 on iains/gcc-15-branch [1], which does not yet have a proper release (15.1 is based on the rc1), so there is no 15.2 version yet.

For the snapshots, we use iains/gcc-darwin-arm64 [2], which is updated regularly.

[1] https://github.com/iains/gcc-15-branch

[2] https://github.com/iains/gcc-darwin-arm64

From: Luke (Lucretia)
Date: Thu, 2 Oct 2025 14:14:26 +0000

Any chance of getting LLVM in Alire for WebAssembly (and other target) support?

From: César Sagaert (csagaert)
Date: Thu, 2 Oct 2025 15:24:04 +0000

There is an existing (but old) PR for GNAT LLVM in GNAT FSF builds, we would like to revisit it at some point.

There are some things that we'll have to make decisions about, like what targets we should have, whether the WASM runtime should be an Alire crate or be bundled with the compiler…

VSS Split into Two Libraries

From: Fernando Oleo Blanco (Irvise)
Subject: VSS has been split into two libraries and main repo is now archived
Date: Sun, 5 Oct 2025 17:21:41 +0000
Newsgroups: forum.ada-lang.io

EDIT: read the post by @jmiven just right below as it gives the actually correct information.

I just became aware that VSS [1] has been archived and the repo has been renamed to reflect such change.

Maybe the Ada-Lang org could create a fork of the archived tools and allow for further community maintenance. After all, most of the tools created by AdaCore are open and we can further develop them if there is a will and action to do so.

[1] https://github.com/AdaCore/VSS-archived

From: Vivien Moreau (jmiven)
Date: Sun, 5 Oct 2025 17:48:08 +0000

There is a deprecation note in the readme, with links to split packages vss-text [1] and vss-extra [2]. I believe those are alive and well?

[1] https://github.com/AdaCore/vss-text

[2] https://github.com/AdaCore/vss-extra

From: Fernando Oleo Blanco (Irvise)
Date: Sun, 5 Oct 2025 18:01:02 +0000

Ouch, I gave it a quick look just in case but I did not see the notice. I will update the title to reflect the change.

Thank you and welcome to the forums!!!

Automatic Styling of Ada Code?

From: reinert (reinert)
Subject: Automatic Styling of Ada Code?
Date: Mon, 6 Oct 2025 04:02:23 +0000
Newsgroups: forum.ada-lang.io

My experimental Ada program now consists of almost 30.000 lines and is becoming less of a personal playground. Before release I should transform it according to a well-known/common style guide. Any hint about how to do this in a quick way - something more than the simplest code formatters and beautifiers?

From: Shuihuzhuan
Date: Mon, 6 Oct 2025 05:39:26 +0000

I use gnatformat --no-subprojects --charset=utf8 (or alr exec -- gnatformat --no-subprojects --charset=utf8 with Alire).

From: Quentin (Heziode)
Date: Mon, 6 Oct 2025 09:21:59 +0000

In addition, you can also add a package inside your GPR to configure gnatformat, ex:

```
project Amazing_Thing is
  -- your config
  package Format is
    for Width ("Ada") use "119";
    -- Max char per line
    for Charset ("Ada") use "utf-8";
    -- default encoding
  end Format;
end Amazing_Thing;
```

You can refer to the official documentation [1] for more details.

Furthermore, you can define a pre-commit hook to format the code automatically before committing. In VS-Code you can also configure Format-On-Save. For GNAT Studio, I don't know.

[1] https://docs.adacore.com/live/wave/gnatformat/html/user-guide/user-guide/configuration.html#the-project-file-attributes

From: OneWingedShark
Date: Mon, 6 Oct 2025 22:50:15 +0000

J.P. Rosen is with a company called AdaLog, and they provide a utility for checking Ada rules (very customizable), and I think that there's a formatter, or at least formatting checker: AdaControl [1].

[1] https://www.adalog.fr/en/adacontrol.html

From: Quentin (Heziode)
Date: Tue, 7 Oct 2025 09:24:15 +0000

J.P. Rosen was my N+1, and he is my company's PhD supervisor. I work for Novasys Ingénierie, where AdaLog is a subsidiary.

Yes, in AdaControl, we have a few "formatting" rules (called Style): AdaControl User Guide > Style [1]

It is possible to apply auto-fix, which acts as gnatformat. *BUT*, AdaControl is firstly a /coding rule verification tool/, not a formatter. So it is not as convenient as gnatformat for formatting.

Furthermore, due to ASIS-4-GNAT end-of-life, AdaControl only supports Ada up to Ada 2012.

[1] https://www.adalog.fr/compo/adacontrol_ug.html#Style

AdaStudio 2026.1

From: Leonid Dulman (AdaStudio)
Subject: Announce: AdaStudio 2026.1
Date: Tue, 7 Oct 2025 07:50:33 +0000
Newsgroups: forum.ada-lang.io

Announce: AdaStudio-2026.1 release 07/10/2025 free edition

It based on Qt-6.10.0-everywhere opensource (expanded with modules from Qt-5.15: qtgamepad, qtx11extras, qtwinextras), VTK-9.5.1, FFMPEG-7.1, OpenCV-4.12.0, Whisper-1.7.6, SDL2-2.24.0, QtAV-1.13

Qt6ada version 6.10.0 open source and qt6base.dll, qt6ext.dll (win64), libqt6base.so, libqt6txt.so (x86-64) built with Microsoft Visual Studio 2023 x64 Windows, GCC amd64 in Linux.

Package tested with GNAT gpl 2020 Ada compiler in Windows 64bit , Linux amd64 Debian 12.

AdaStudio-2026.1 includes next modules: qt6ada, vtkada, qt6mdkada, qt6cvada (face recognition, face detection, face identification, objects detection, QRcode detector, BARcode detection and others) and voice recognizer.

Qt6Ada is built under áGNU LGPLv3álicense GNU Lesser General Public License v3.0 - GNU Project - Free Software Foundation [1] .

Qt6Ada modules for Windows, Linux (Unix) are available from

Google drive AdaStudio - Google Drive [2]

WebPage is https://r3fowwcolhrzycn2yzlzzw-on.drv.tw/AdaStudio/adastudio.html

[...]

Ada 2022 Hybrid Reference Architecture

From: Michael Gardner (abitofhelp)
Subject: Ada 2022 Hybrid Reference Architecture
Date: Wed, 29 Oct 2025 10:49:12 +0000
Newsgroups: forum.ada-lang.io

I'm excited to announce the release of the Ada 2022 Hybrid Reference Architecture—a comprehensive project that includes a reference codebase and a 240-page primer on implementing DDD, Clean Architecture, and Hexagonal patterns using generics instead of OOP interfaces. The primer textbook is complete and currently under editorial review—stay tuned!

If you've been searching for "hexagonal clean architecture with generics", you've found it!

This isn't just an Ada project—it's a mental model for any language that favors generics over interfaces. The patterns translate directly to Rust, modern C++, and beyond.

Complete working implementation - Production-ready Ada 2022 code ready to use as a starter template.

Comprehensive primer textbook - Mental model translation from OOP to generics

Side-by-side UML diagrams - Compare the same patterns in both paradigms visually

Compiler-enforced boundaries - Architecture violations literally won't compile

Professional starter template - Clone and adapt for your own hexagonal projects

Unique Features:

- Three-layer architecture enforcement (compiler + validation script + CI/CD)
- Application.Result facade pattern (breaks Presentation→Domain coupling elegantly)
- Bounded strings at API boundaries (predictable memory, zero allocation failures)
- 10 comprehensive test suites - all passing, zero violations

GitHub: https://github.com/abitofhelp/hybrid_ada

Full Documentation: Complete with architecture guides, testing strategies, and detailed explanations

Functional Error Handling v1.0.0

From: Michael Gardner (abitofhelp)
Subject: Functional Error Handling v1.0.0: Result<t,e>, Option<t>, and Either<l,r>, Try_to_result
Date: Wed, 29 Oct 2025 22:52:03 +0000
Newsgroups: forum.ada-lang.io

I'm happy to announce my latest open-source Ada 2022 project: "Functional v1.0.0"!

Github: Functional Project [1]

Alire: Functional Crate [2]

Features:

- Result - Type-safe error handling (17 operations)
- Option - Optional values (11 operations)
- Either - Disjoint union type (8 operations)
- Try.To_Result - Convert exceptions to Result
- Try.To_Option - Convert exceptions to Option
- Pure packages - No side effects, compile-time guarantees
- Zero dependencies - Just Ada 2022 standard library
- Production-ready - Comprehensive compiler checks and style enforcement

[1] https://github.com/abitofhelp/functional

[2] https://alire.ada.dev/crates/functional.html

PolyORB End of Life

From: Rod Kay (charlie5)
Subject: PolyORB end of life
Date: Fri, 31 Oct 2025 20:47:19 +0000
Newsgroups: forum.ada-lang.io

"End of Life Notification

PolyORB is a deprecated product. It will be baselined with the GNAT Pro release 28. After this release, there will be no new versions of this product. Contact your sales representative or send a message to sales@adacore.com [1] to get recommendations for replacements."

Sad to say but PolyORB is now officially dead. This leaves Ada without any DSA support, which hits home for me at least, since several of my projects rely heavily on DSA. I've been using PolyORB since its inception, when it used customised Makefiles instead of po_gnatdist. So many, many years of wasted development.

I had thought, if this happened, it might be possible to fork PolyORB, strip it down to only the DSA personality and continue to maintain it. However, I doubt this is feasible and am pretty certain I lack the skill. It would take a 1/2 dozen or so dedicated developers with enough time on their hands, at least.

Seems a shame DSA is only an annex and not a part of the core Ada spec but I expect that would impose too heavy a burden on compiler developers who have to cope with what is already a very complex, yet magnificent, language.

[1] mailto:sales@adacore.com

From: Michał (pmnw)
Date: Fri, 31 Oct 2025 21:08:11 +0000

> Seems a shame DSA is only an annex and not a part of the core Ada spec but I expect that would impose too heavy a burden on compiler developers who have to cope with what is already a very complex, yet magnificent, language.

It seems ARG is aware more work is needed in the distributed area.

Perhaps the following can give you some peace of mind.

> Better support for heterogeneous and distributed environments

> Modern computing often involves systems of various architectures, connected either with shared memory or over a high-speed bus. Ada's Distributed System Annex is one approach, but has not been widely implemented nor used. A "building block" approach might be one alternative way to support heterogeneous and distributed systems. This is clearly related to the above item related to representing arbitrary types in some sort of stream representation.

Instructions to the Ada Rapporteur Group from SC22/WG9 [1]

https://docs.google.com/document/d/1w2fbtFeBqWxOelhY4iYvRCgLjrt3S7cpmZjelwWv2A/edit

From: Tucker Taft (sttaft)
Date: Sat, 1 Nov 2025 01:20:32 +0000

It would be helpful to create an ARG GitHub Issue about distributed systems support in Ada going forward, and try to capture the pros and cons of the DSA approach. Please feel free to create the issue here:

https://github.com/Ada-Rapporteur-Group/User-Community-Input/issues

From: Rod Kay (charlie5)
Date: Fri, 7 Nov 2025 22:44:38 +0000

I've added an issue here … Modernisation of the Distributed Systems Annex (DSA)

https://github.com/Ada-Rapporteur-Group/User-Community-Input/issues/153

Although I expect Dmitri would have done much better. It's based on ChatGPT response to "pros/cons of DSA?" plus a little from an old 'comp.lang.ada' thread.

Interest in Binaries for GNAT-LLVM

From: Matheus Xavier (xadaemon)
Subject: Is there interest in prebuilt binaries for the GNAT-LLVM?
Date: Sat, 8 Nov 2025 20:28:50 +0000
Newsgroups: forum.ada-lang.io

I'm trying to figure out a pipeline to build gnat-llvm reliably and I was wondering if there is interest for these binaries made public, and maybe added to Alire (I would be willing to work to make the builds reproducible to help establish trust), I mainly want to make it easier to experiment with Ada in WASM.

From: Max Reznik (Max)
Date: Sat, 8 Nov 2025 20:38:58 +0000

There's draft PR to build it for Linux:

GNATLLVM for WASM32 target by godunko

https://github.com/alire-project/GNAT-FSF-builds/pull/83

TZif v1.0.0

From: Michael Gardner (abitofhelp)
Subject: Tzif v1.0.0: Finally! Iana Timezone Database Parser and Query Engine for Ada 2022!
Date: Fri, 14 Nov 2025 09:20:24 +0000
Newsgroups: forum.ada-lang.io

Thrilled to announce TZif v1.0.0 was published to Alire today and should be available soon!

TZif delivers trusted IANA-compiled timezone access and querying for Ada 2022! Built on principles of domain-driven design, clean & hexagonal architecture, and Ada 2022 contracts, TZif brings the best practices of modern functional programming languages to the Ada ecosystem.

Battle-Tested – 252 passing tests prove it's ready for production

Zero Exceptions – Pure functional error handling with Result monads means no runtime surprises

Type-Safe – Private value objects with validated constructors catch errors at compile time

Railway-Oriented – Elegant error propagation that makes your code cleaner and safer

Cache-Ready – JSON export/import for lightning-fast startup times

Cross-Platform – Works seamlessly on Linux, macOS, BSD, and Windows (partial, full v1.1.0)

13 working examples included to get you started in minutes!

RFC 9636 TZif parser with functional error handling

https://github.com/abitofhelp/tzif_ada

Zed Ada Language v0.4.0

From: wisn
Subject: Zed Ada Language v0.4.0 Release
Date: Mon, 17 Nov 2025 15:41:21 +0000
Newsgroups: forum.ada-lang.io

We released Zed Ada Language [1] v0.4.0 yesterday and have merged it into the Zed extensions repository today.

This release introduces language support for GPR files and adds local symbol navigation functionality.

All of these changes are thanks to our first contributor, Sebastian Poeplau [2]!

Preview attached below. I used Septum for the preview.

The first screenshot shows local symbol navigation. The second screenshot shows language support for GPR files.

Any feedback is highly appreciated. Thank you!

[1] https://github.com/wisn/zed-ada-language

[2] https://github.com/sebastianpoeplau

From: Jon (docandrew)
Date: Mon, 17 Nov 2025 17:29:38 +0000

Starting to experiment with Zed - will check this out!

Using OpenAI API from Ada

From: reinert (reinert)
Subject: Anybody Tried Openai-api from Ada?
Date: Mon, 1 Dec 2025 08:36:52 +0000
Newsgroups: forum.ada-lang.io

Is there any experience out there with using the OpenAI API from Ada? Does anyone have anything to share?

From: Stephane Carrez (stcarrez)
Date: Wed, 3 Dec 2025 20:48:27 +0000

I've played a little with that last year and created an Alire crate. The OpenAI API requires a token that you obtain from your account. Their API seems to change quickly so I'm not sure this is still up to date.

Stephane Carrez / ada-openai · GitLab [1]

And you will find examples in for image generation: [2]

And chat operation: [3]

GitLab.com

[1] https://gitlab.com/stcarrez/ada-openai

[2] https://gitlab.com/stcarrez/openai-image

[3] https://gitlab.com/stcarrez/openai-chat

From: Max Reznik (Max)
Date: Sat, 6 Dec 2025 21:05:56 +0000

Just tried it with ollama to access local running models and it works fine! Thank you!

CoAP-SPARK v0.10.0

From: Manuel (mgrojo)
Subject: Coap-spark Version 0.10.0 Has Been Published
Date: Tue, 2 Dec 2025 22:53:54 +0000
Newsgroups: forum.ada-lang.io

CoAP-SPARK is a library implementing the Constrained Application Protocol (CoAP) as defined in RFC 7252 [1], developed in SPARK and formally verified.

The version 0.10.0 has been published in the Alire community index. The new release adds server support and a sample server implementation.

Alire - Coap_spark [2]

Release v0.10.0 · mgrojo/coap_spark [3]

What's Changed

CoAP server support in the library and example program by @mgrojo in #17
Update macOS version in CI workflow by @mgrojo in #16

[1] https://www.rfc-editor.org/rfc/rfc7252

[2] https://alire.ada.dev/crates/coap_spark

[3] https://github.com/mgrojo/coap_spark/releases/tag/v0.10.0

Strings Edit v3.10

From: Dmitry Kazakov (dmitry-kazakov)
Subject: Strings Edit v3.10
Date: Thu, 4 Dec 2025 15:28:16 +0000
Newsgroups: forum.ada-lang.io

The library provides string handling facilities like I/O formatting, Unicode and obsolete code pages support.

String processing for Ada [1]

This update provides full Unicode normalization support. Unicode normalization is intended to equalize same looking glyphs in order to compare them. In particular normalization applies to diacritic marks like ü = u + ◌̈, ligatures like fi = fi, symbols like Ω = Ohm symbol, subscripts, superscripts. However normalization does not apply to Latin and Cyrillic letters nor ligatures like German ß.

Changes to the previous version:

- Function Compare to compare arrays of code points was added to the package Strings_Edit.UTF8;
- Function Image to convert a code points array to UTF-8 string was added to the package Strings_Edit.UTF8;
- Function Compare to compare arrays of code points using a code points mapping was added to the package Strings_Edit.UTF8.Maps;
- The package Strings_Edit.UTF8.Normalization was added to provide Unicode decompositions (NFD and NFKD), composition, normalization (NFC, NFKC) as well as comparisons of normalized strings. The canonical composition rules are respected;
- The application Strings_Edit.UTF8.Normalization_Generator was added to support updates of the UnicodeData.txt data base;
- The test case test_utf8 was added.

[1] https://www.dmitry-kazakov.de/ada/strings_edit.htm

Simple Components v4.77

From: Dmitry Kazakov (dmitry-kazakov)
Subject: Simple Components v4.77
Date: Fri, 5 Dec 2025 08:24:30 +0000
Newsgroups: forum.ada-lang.io

The current version provides implementations of smart pointers, directed graphs, sets, maps, B-trees, stacks, tables, string editing, unbounded arrays, expression analyzers, lock-free data structures, synchronization primitives (events, race condition free pulse events, arrays of events, reentrant mutexes, deadlock-free arrays of mutexes), arbitrary precision arithmetic, pseudo-random non-repeating numbers, symmetric encoding and decoding, IEEE 754 representations support, streams, persistent storage, multiple connections server/client designing tools and protocols implementations.

Simple components for Ada [1]

This update is focused on improving Python bindings.

Changes the previous version:

- The Python high-level class support fixed to accommodate controlled types;
- The Python high-level class supports extension of Python types with Ada type extension;
- Binding to Get_Basic_Size were added to the package Py;
- Throw_NotImplementedError procedure was added to the package Py;
- Weak references support was added to the Py package.

[1] https://www.dmitry-kazakov.de/ada/components.htm

Functional Programming Library for Ada 2022

From: Michael Gardner (abitofhelp)
Subject: Functional Programming Library for Ada 2022
Date: Sat, 6 Dec 2025 04:44:59 +0000
Newsgroups: forum.ada-lang.io

A production-ready Ada 2022 library providing functional programming abstractions for type-safe computation.

Implements Result, Option, and Either monadic types with 87 composable operations enabling railway-oriented programming, explicit error handling, and optional value management.

Designed for safety-critical, embedded, and high-assurance applications with full SPARK compatibility.

https://github.com/abitofhelp/functional

From: Michael Gardner (abitofhelp)
Date: Mon, 8 Dec 2025 00:00:23 +0000

v3.0.0 Released: 100% SPARK Proof Verification!

The Functional Ada 2022 library v3.0.0 now has 100% SPARK proof verification with zero unproved checks.

Results

[...]

What's Verified

- Option[T] - 26 operations with postconditions
- Result[T,E] - 36 operations with postconditions
- Either[L,R] - 20 operations with postconditions

Guarantees

- No runtime errors (overflow, range violations)
- No uninitialized data access
- All pre/postconditions proven correct
- Termination proven for all functions

[...]

SPARK verification uses test instantiations since gnatprove analyzes instantiated generics, not templates.

IANA Timezone Information File Library for Ada 2022

From: Michael Gardner (abitofhelp)
Subject: IANA timezone information file library for Ada 2022
Date: Mon, 8 Dec 2025 07:54:02 +0000
Newsgroups: forum.ada-lang.io

TZif is an Ada 2022 library for parsing and querying IANA's compiled timezone information (TZif format, RFC 9636) files. It provides a clean, functional API with Result monad error handling, hexagonal architecture, and embedded-safe patterns.

SPARK Formal Verification

Designed for safety-critical, embedded, and high-assurance applications with full SPARK compatibility.

Status SPARK Proved

Scope Domain + Application layers (value objects, containers, parser, operations, ports)

Mode gnatprove --mode=prove --level=2

Results 1350 checks: 1155 proved, 195 unproved (in generic instantiations)

Features

- Parse IANA TZif binary files (versions 1, 2, and 3)
- Query timezone transitions at any Unix epoch time
- Discover and validate timezone data sources
- Find zones by ID, pattern, region, or regex
- Detect the system's local timezone
- Cross-platform: Linux, macOS, BSD, Windows 11, Embedded
- 4-layer hexagonal architecture (Domain, Application, Infrastructure, API)
- Result monad error handling (via functional crate)
- Generic I/O plugin pattern for platform portability

https://github.com/abitofhelp/tzif_ada/

Hybrid DDD/Clean/Hexagonal Library Starter for Ada 2022

From: Michael Gardner (abitofhelp)
Subject: Hybrid DDD/clean/hexagonal library starter for Ada 2022
Date: Wed, 10 Dec 2025 08:18:57 +0000
Newsgroups: forum.ada-lang.io

Overview

Hybrid_Lib_Ada is a professional Ada 2022 starter library demonstrating a hybrid DDD/Clean/Hexagonal architecture with functional error handling.

Key Capabilities:

- 4-layer hexagonal architecture (Domain, Application, Infrastructure, API)
- Functional error handling via Result monad (no exceptions)
- Three-package API pattern for flexible dependency injection
- Generic I/O plugin pattern for platform portability
- Embedded-safe design (no heap allocation, bounded types)
- SPARK-compatible for formal verification (6 proofs verified)
- Cross-platform: Linux, macOS, Windows, Embedded

https://github.com/abitofhelp/hybrid_lib_ada/

[Reminder] The PragmAda Reusable Components

From: Pragmada Software Engineering <pragmada@pragmada.x10hosting.com>
Subject: [Reminder] The PragmAda Reusable Components
Date: Wed, 10 Dec 2025 09:24:14 +0000
Newsgroups: comp.lang.ada

The PragmARCs are a library of (mostly) useful Ada reusable components provided as source code under the GMGPL or BSD 3-Clause license at https://github.com/jrcarter/PragmARC.

This reminder will be posted about every six months so that newcomers become aware of the PragmARCs. I presume that those who want notification when the PragmARCs are updated have used Github's notification mechanism to receive them, so I no longer post update announcements. Anyone who wants to receive notifications without using Github's mechanism should contact me directly.

IANA Timezone DateTime Manipulation Library for Ada 2022

From: Michael Gardner (abitofhelp)
Subject: IANA timezone datetime manipulation library for Ada 2022
Date: Wed, 17 Dec 2025 03:45:06 +0000
Newsgroups: forum.ada-lang.io

GitHub [1]

Alire [2]

Overview

Zoneinfo is an IANA timezone datetime manipulation library for Ada 2022. Built on the TZif library for IANA database access, it provides a clean, type-safe API for working with timezones, parsing ISO 8601 strings, formatting datetimes, and discovering timezone sources. The library follows hybrid DDD/Clean/Hexagonal architecture with functional error handling and is designed for both desktop and embedded platforms.

Features

- Timezone-Aware Datetimes - Instant, Zoned, and Civil time representations
- ISO 8601 Parsing - Parse datetime strings with timezone offsets and zone IDs
- Datetime Formatting - Format datetimes as ISO 8601 or custom patterns
- Timezone Discovery - Find system timezone, search by pattern/region/regex (bounded arrays)
- Duration Arithmetic - Add/subtract durations, calculate differences
- TZif Integration - Built on the TZif library for IANA timezone database access
- Result Monad Error Handling - No exceptions, functional error handling
- 4-Layer Hexagonal Architecture - Domain → Application → Infrastructure → API
- SPARK Verified - Domain + Application layers formally verified
- Embedded Safety - No implicit heap allocations, bounded types, static dispatch
- Library Standalone - Explicit Library_Interface for ABI stability

SPARK Formal Verification [...]

See CHANGELOG for current verification statistics

[1] https://github.com/abitofhelp/zoneinfo_ada

[2] https://alire.ada.dev/crates/zoneinfo.html

Gerrit Will Soon Support Ada Syntax Highlighting

From: Manuel Stahl (ThyMYthOS)
Subject: Gerrit Will Soon Support Ada Syntax Highlighting
Date: Sun, 21 Dec 2025 09:44:09 +0000
Newsgroups: forum.ada-lang.io

See https://gerrit-review.googlesource.com/c/gerrit/+/537881

AoR JSONA - Typed Extraction from JSON Payloads

From: David Sauvage - AdaLabs Ltd (david.sauvage)
Subject: AoR JSONA - Typed extraction from JSON payloads via path-based, indexed navigation
Date: Wed, 24 Dec 2025 10:03:20 +0000
Newsgroups: forum.ada-lang.io

Announce: AOR JSONA - Typed extraction from JSON payloads via path-based, indexed navigation

https://gitlab.com/adalabs/aor_jsona

Jsona is a small Ada on Rails component that enables typed extraction from JSON payloads via path-based, indexed navigation.

https://alire.ada.dev/crates/aor_jsona

SweetAda on Veecom

From: Gabriele Galeotti (SweetAda)
Subject: SweetAda on Veecom
Date: Sun, 28 Dec 2025 16:15:12 +0000
Newsgroups: forum.ada-lang.io

SweetAda [1] “is a lightweight development framework whose purpose is the implementation of Ada-based software systems”.

Veecom (GitHub - Mazin-O3/Veecom: Simple Yet Powerful RISC-V Computer [2]) “is a 32-bit computer system that combines the simplicity of 8-bit home computers with modern RISC-V technology”. Veecom is a bare digital hardware Logisim-evolution design.

Just for the hardware-cored users, SweetAda, as of commit bbc8ed7, now supports Veecom with no changes to the underlying code base. If you want to see Ada code moving electrons on wires by means of the Logisim simulator, you can try it. Veecom so far lacks exception support, but it simulates an RV32IM CPU, along with a simple DMA controller and a VIA I/O device which interfaces to a TTY console, everything inside the digital simulator.

In order to run SweetAda on Veecom, you need the SRecord package (https://srecord.sourceforge.net/), which converts the final binary code to a srec-compatible format. The target “make

postbuild" runs a postbuild.sh script (only *nix, sorry I still do not have so far Logisim on a Wins machine, but creating a .bat should be extremely simple) that generates a suitable memory image. (PS: anyway, I am working in removing the need by using a little Python/Tcl script)

Once the project is loaded inside Logisim, go into the memory block and right-click to load the memory image, then activate clock ticks to start the CPU.

After a long time (depending on your machine power, maybe just a minute), you should be able to see the welcome message on the simulated TTY console.

[1] https://github.com/gabriele-galeotti/SweetAda

[2] https://github.com/Mazin-O3/Veecom

Ada-related Products

Updated List of Ada Compilers

[Continuation of a thread where compilers other than GNAT are discussed, that did not appear in previous AUJ issues. —arm]

From: Laurent
Subject: Updated List of Ada Compilers
Date: Tue, 28 Oct 2025 16:33:37 +0000
Newsgroups: forum.ada-lang.io

DDC-I still supplies Ada as well.

From: Manuel (mgrojo)
Date: Tue, 28 Oct 2025 21:47:42 +0000

I recently removed XGC from the awesome-ada list [1], because the website has disappeared and haven't found any trace of them, so I assume they've closed. If someone knows that they have just moved, please, open a pull request.

> DDC-I still supplies Ada as well.

Should that be added to the awesome-ada list?

[1] https://github.com/ohenley/awesome-ada

From: Dirk Craeynest (DirkCraeynest)
Date: Thu, 30 Oct 2025 07:23:36 +0000

> I recently removed XGC from the awesome-ada list, because the website has disappeared and haven't found any trace of them, so I assume they've closed.

XGC Technology is/was Chris Nettleton. Unfortunately, Chris died on June 4th 2025. Not sure what happens with the tools he supported…

From: Ada Orbit (AdaOrbit)
Date: Tue, 28 Oct 2025 22:27:43 +0000

There are still a number of old Ada compilers being licensed. I don't know when and where I found this but here is one of them [1]. They only have support for Ada83 and Ada95 tho. (2005 edition is still in beta)

[1] http://irvine.com/products.html

Ada and Operating Systems

What Is the State of Ada on Android?

From: mekalu
Subject: What Is the State of Ada on Android?
Date: Sat, 13 Dec 2025 12:33:23 +0000
Newsgroups: forum.ada-lang.io

I am building a graphics and audio engine (In Zig) and for a while I wanted to port some of my code to Ada but one of my targets is Android and GCC does not support it. It is unfortunately a major target for me.

I found out that there is an LLVM front-end but what is the state of it? I cannot really find relevant information. Has anyone tried to target Android with it? Or is it still experimental?

I also found that there is some sort of android compiler ported to bsd [1] but I have no idea how well it works or even maintained anymore…

[1] https://decijferij.nl/software/Ada on Android.html

From: nerd_type
Date: Sun, 14 Dec 2025 00:24:33 +0000

This is just an idea, but it might be possible to build GNAT LLVM (targeting the desired Android architecture) using the Android NDK (as a sysroot). Troy has identified a working configuration of GNAT LLVM [1] you could start with.

With an LLVM compiler that can generate C and Ada code on the correct platform you could try to build the "rawdrawandroid" project by cnlohr [2]. This brilliant project demonstrates building Android applications using just C and Makefiles (no Java).

At this point you will have a project that runs on the Android platform and a compiler that can generate Ada code. Continuing on, in theory you could create an Ada run time that integrates with the Android framework and use it as the basis for Ada application development on Android.

[1] https://github.com/brownts/build-gnat-llvm

[2] https://github.com/cnlohr/rawdrawandroid

From: Kohlrak
Date: Sun, 14 Dec 2025 06:44:25 +0000

I'm kinda curious about the state of the termux thing, too. I know Android was trying to restrict things like Termux from creating and running executables, but that was over a year ago and you can still download termux. I remember using that tutorial (and a few others) and compiling an example app for Android /ON/ Android.

From: Fernando Oleo Blanco (Irvise)
Date: Sun, 14 Dec 2025 09:38:02 +0000

AFAIK, support for arm64 and Android has been improved in the latest versions of GCC (as per the changelogs). Also, as already pointed out, there is LLVM-19 support for GNAT-LLVM, which should be plenty flexible to get Android stuff running with Ada.

It is true that in the past there was more "easy to get started" support, but not anymore. But that does not mean that the behind the scenes support has not been improving!

References to Publications

Ada Books

[Note that those books have not been vetted for quality and, these days, AI slop is always a risk, especially with self-published editions. —arm]

W. Smith, "Mastering Ada Programming: From Basics to Expert Proficiency", HiTeX Press, Jun. 14, 2025. Available: https://www.kobo.com/us/en/ebook/mastering-ada-programming. (EPUB2; ISBN/EAN: 6610000612734).

T. Edet, "Ada Programming: Reliable, Strongly-Typed Systems Programming", Amazon Digital Services LLC (Kindle Direct Publishing), Jun. 7, 2025. Available: https://www.amazon.com/-/en/Ada-Programming-Strongly-Typed-Mastering-Languages/dp/B0FCD817JR. (Print edition; ISBN-13: 979-8287237165).

Old but Cool: Ada Mapping to CORBA

From: OneWingedShark
Subject: Old but cool: Ada mapping to CORBA
Date: Fri, 3 Oct 2025 23:02:38 +0000
Newsgroups: forum.ada-lang.io

Kinda-sorta relevant COM/DCOM in Ada: Interfacing Ada 95 to Microsoft COM and DCOM Technologies [1].

(Active COM: an inter-working framework for CORBA and DCOM [2].)

[1] https://dl.acm.org/doi/pdf/10.1145/319294.319297

[2] https://ieeexplore.ieee.org/abstract/document/794029

NVIDIA Publishes SPARK Process to Meet ISO-26262 Requirements

From: Fernando Oleo Blanco (Irvise)
Subject: NVIDIA publishes SPARK process to meet ISO-26262 requirements
Date: Wed, 22 Oct 2025 18:04:13 +0000
Newsgroups: forum.ada-lang.io

Their newer edition with updates is https://www.youtube.com/watch?v=KhWtkZmOPn4

[Article] Ada: Should Developers Revisit the Veteran Programming Language?

From: dragon-spirit-wtp (dragon-spirit-wtp)
Subject: (article) Ada: Should Developers Revisit the Veteran Programming Language?
Date: Fri, 14 Nov 2025 18:02:52 +0000
Newsgroups: forum.ada-lang.io

Developer Tech News – 13 Nov 25 [1]

> Ada: Should developers revisit the veteran programming language?

> Ada, a 45-year-old programming language, might just solve the very problems developers have been grappling with for years.

> Estimated reading time: 4 minutes

[1] https://www.developer-tech.com/news/ada-should-developers-revisit-veteran-programming-language/

Ada User Society Yearly (2025) Bulletin

From: Fernando Oleo Blanco (Irvise)
Subject: Ada User Society Yearly (2025) Bulletin
Date: Tue, 30 Dec 2025 20:46:11 +0000
Newsgroups: forum.ada-lang.io

For those who are not members (yet?) of the Ada User Society, you can read the 2025 Yearly Bulletin in the website Ada User Society Yearly Bulletin [1]

Happy new year to all!

[1] https://www.ada-user.org/ada/user/society/news/2025/12/17/ada-user-yearly-bulletin.html

From: geewiz
Date: Tue, 30 Dec 2025 21:12:51 +0000

Thanks Fer, you just inspired me to join both Ada User Society and Ada Europe as a member.

Ada Inside

Configurable Bareboard Tasking Runtimes for RP2040

From: damaki
Subject: Configurable bareboard tasking runtimes for RP2040
Date: Mon, 6 Oct 2025 20:06:13 +0000
Newsgroups: forum.ada-lang.io

I have just released new light-tasking and embedded runtime crates for the RP2040 [1].

Ada/SPARK runtimes for the Raspberry Pi RP2040

What sets these apart from the Pico runtimes already distributed in gnat_arm_elf is that these runtimes are configurable via Alire's crate configuration [2], the purpose being to make the runtime easier to tweak without needing to dig into the runtime sources (which isn't always the friendliest of experiences). In particular, the following aspects are currently configurable via the crate config:

- the number of Cortex-M0+ cores used by the runtime (1 or 2 cores);
- which alarm IRQ is used by the runtime for task delays (useful for avoiding clashes if the alarm IRQ is needed for something else);
- which board the runtime is configured for (e.g. the Pico, Pimoroni boards, AdaFruit boards, etc);
- the clock configuration, so you can change the frequency at which the system runs; and
- the interrupt stack sizes.

The cool thing about using the tasking runtimes is that it's dead easy to write multicore programs for the Pico, since tasks can be pinned to a specific core using the CPU pragma/aspect. For example, to declare a task which runs on the second core:

```
task My_Task with CPU => 2;
```

See the GitHub link above for more info.

[1] https://github.com/damaki/rp-runtimes/

[2] https://alire.ada.dev/docs/#using-crate-configuration

Bareboard Tasking Runtimes for the RP2350 (Raspberry Pi Pico 2)

From: damaki
Subject: Bareboard tasking runtimes for the RP2350 (Raspberry Pi Pico 2)
Date: Sat, 25 Oct 2025 18:35:03 +0000
Newsgroups: forum.ada-lang.io

I finally got my hands on a Pico 2, so I've been able to release runtime crates for the RP2350. The new crates are:

- light_tasking_rp2350 [1]
- embedded_rp2350 [2]

Like the RP2040 crates I announced previously [3], these crates are also configurable through crate configuration variables. See the readmes on the GitHub page [4] for more info on configuring them (in particular, README-RP2350.md)

For the moment the runtimes only support the Cortex-M33 cores. I might explore the RISC-V cores later if there's demand for it.

[1] https://alire.ada.dev/crates/light_tasking_rp2350

[2] https://alire.ada.dev/crates/embedded_rp2350

[3] https://forum.ada-lang.io/t/ann-configurable-bareboard-tasking-runtimes-for-rp2040/3888

[4] https://github.com/damaki/rp-runtimes

From: Nicolas Pinault (DrPi)
Date: Sat, 25 Oct 2025 20:46:57 +0000

Thanks for your work. That's very useful.

I don't really understand the difference between both runtimes. It seems to be tasking related. But what are these differences?

From: damaki
Date: Sat, 25 Oct 2025 21:34:19 +0000

"light tasking" and "embedded" are two pre-defined runtime profiles for GNAT, which provide different levels of language support. For bareboard targets, there are generally three different profiles that can be supported:

- "light" runtimes are the smallest runtimes but lack support for some Ada features. In particular, they don't support tasking or exception propagation, and have a simple heap allocator (Unchecked_Deallocate is a no-op). They do, however, provide support for things like assertions, 'Image and 'Value attributes for scalar types, simplified Ada.Text_IO, secondary stack.
- "light-tasking" runtimes include everything in "light", but also add support for a tasking subset based on the Ravenscar and Jorvik profiles. For these RP2350 runtimes, this means you can declare multiple tasks and assign them to either of the two Cortex-M33 cores. For example, task My_Task with CPU => 2; declares a task that will run on the second core of the RP2350. You can also use Ada's delay until statement for task delays, and use protected objects for inter-task communication and for interrupt handlers.
- "embedded" runtimes include everything in "light-tasking", but also

adds support for exception propagation and has a full heap allocator (so you can use Unchecked_Deallocate to free memory).

For more information on what's supported in each runtime, see this part of the GNAT User's Guide Supplement for Cross Platforms: 5. Predefined GNAT Pro Run-Times — GNAT User's Guide Supplement for Cross Platforms 27.0w documentation [1]

I hope this helps. Happy to answer any more questions if something isn't clear.

[1] https://docs.adacore.com/live/wave/gnat_ugx/html/gnat_ugx/gnat_ugx/gnat_runtimes.html#

AArch64 Bare-Metal Ada Library

From: J. German Rivera (jgrivera67)
Subject: aarch64_bare_metal_ada: A library crate for developing bare-metal embedded software in Ada for 64-bit ARM platforms
Date: Wed, 29 Oct 2025 21:05:07 +0000
Newsgroups: forum.ada-lang.io

I have released version 1.0.0 of the aarch64_bare_metal_ada library crate [1]. This crate contains a collection of bare-metal SPARK Ada software components that can be used as an infrastructure to develop bare-metal multicore system-level software in Ada, for AArch64-based embedded platforms. It can be used as a foundation to develop boot ROMs, boot loaders, multicore RTOS kernels, separation kernels and hypervisors in Ada, for AArch64-based platforms.

A tutorial that teaches how to develop 64-bit bare-metal embedded software in Ada for Raspberry Pi 4 and Raspberry Pi 5, using components from this crate, is included here [2].

[1] https://alire.ada.dev/crates/aarch64_baremetal_ada

[2] https://github.com/jgrivera67/aarch64_bare_metal_ada/blob/main/docs/tutorial.md

Airbus A320 Software Update

From: SBP (Seirios)
Subject: [news] Airbus A320 Software Update
Date: Sat, 29 Nov 2025 14:12:03 +0000
Newsgroups: forum.ada-lang.io

Airbus – 28 Nov 25 [1]

Airbus update on A320 Family precautionary fleet action [1]

Analysis of a recent event involving an A320 Family aircraft has revealed that intense solar radiation may corrupt data critical. Read more.

aviationweek.com [2]

EASA Orders Immediate Airbus A320 Flight Control Software Changes [2]

Airbus warns A320 operators that solar radiation could corrupt flight control inputs, prompting EASA to issue an emergency airworthiness directive.

Just saw this and got curious. Does anyone here know about this software update and what it entails? Is Ada involved?

[1] https://www.airbus.com/en/newsroom/press-releases/2025-11-airbus-update-on-a320-family-precautionary-fleet-action

[2] https://aviationweek.com/air-transport/safety-ops-regulation/easa-orders-immediate-airbus-a320-flight-control-software

From: Jeremy Grosser (JeremyGrosser)
Date: Sat, 29 Nov 2025 17:43:44 +0000

Here's the EASA EAD [1], FAA EAD [2] and the AIRBUS AOT [3].

AVHerald has a good summary of the incident [4], excerpts:

> JetBlue Flight 1230, operating from Cancún International Airport (CUN) to Newark Liberty International Airport (EWR), experienced an uncommanded drop in altitude approximately one hour after departure. The aircraft, registered N605JB , rapidly lost about 14,500 feet in five minutes, followed by another 12,200 feet in the next five minutes. The crew diverted to Tampa International Airport (TPA) and landed at approximately 1420 local time.

> On Nov 7th 2025 the NTSB reported: "During cruise, the aircraft experienced an uncontrolled descent for approximately 4-5 seconds before the autopilot corrected the trajectory. This likely occurred during an ELAC switch change." The occurrence caused 10 injuries on board, the NTSB opened an investigation.

> The subsequent investigation identified a vulnerability with the ELAC B hardware fitted with software L104 in case of exposure to solar flares.

I found this paper AIRBUS FLY-BY-WIRE: A TOTAL APPROACH TO DEPENDABILITY [5], which contains some detail about the redundant setup of the ELACs. There's also a Boeing paper [6] referencing that one and comparing it to their implementation.

[1] https://ad.easa.europa.eu/blob/EASA_AD_2025_0268_E.pdf/EAD_2025-0268-E_1

[2] https://drs.faa.gov/browse/excelExternalWindow/DRSDOCID1701465859202511290342 43.0001

[3] https://avherald.com/files/AOT-A27N022-25-00.pdf

[4] https://avherald.com/h?article=52f1ffc3&opt=0

[5] https://link.springer.com/content/pdf/10.1007/978-1-4020-8157-6_18.pdf

[6] https://link.springer.com/content/pdf/10.1007/978-1-4020-8157-6_19.pdf

From: Frédéric Loyer (F-Loyer)
Date: Mon, 1 Dec 2025 23:43:51 +0000

I guess the 4-5 seconds are the time needed to switch to another computer (an eternity if the plane is uncontrolled). Then what should be the main way to avoid solar eruption issues ? (I guess the difference between L104 and L103+ version)

From: nbjessen
Date: Tue, 2 Dec 2025 18:33:54 +0000

I've designed for SEU before. To be safe, you need to take a layered approach: design your HW so single-bit-flips are detected (and corrected if at all possible). So you correct /and/ report that there was an Event in a couple of nanoseconds (literally - it's what we HW people do). Then you report the failure up to the SW system. You have redundant HW (DO-254 DAL-B/DAL-C and DO-178 DAL-B/DAL-C for FAA; I forget the EU's equivalents), and the SW monitors all the redundant HW and if it detects variation or reports of uncorrectable errors, it reboots the failing system and flies "single point of failure" doing the reboot.

In other words - airlines only design for single-point-of-failure. Not more than one. So if one computer gets hit and is in reboot, and then the other reports a failure and has uncorrectable data corruption - you have no source of truth. Or if the governing SW itself gets zapped, how does it know?

Imagine if every time you go to run an opcode, you need to have error-correction logic running, because a cosmic ray might have flipped the bit…

I did this type of work for some years.

From: Brian Drummond (Brian-Drummond)
Date: Sun, 7 Dec 2025 11:23:59 +0000

Just in case anyone is looking for redundant hardware, or wonders what's involved, Texas Instruments offers two families of dual redundant ARM processors, as the TI Hercules series. Both CPUs execute the same code, but 2 clock cycles out of step, with a comparator between the CPUs.

An SEU would affect one of the CPUs; a power glitch or EMI may affect both but on different instructions; in either case the comparator should trigger (and IIRC, raise an interrupt). Logging and recovery are then down to software.

One family is automotive oriented (CANbus etc), the other is oriented towards medical equipment. IMO both would make good targets for Ada in embedded systems…

From: Nicolas Pinault (DrPi)
Date: Sun, 7 Dec 2025 16:27:41 +0000

In the ARM world, this is called lockstep CPUs. Several manufacturers sell this kind of chips (NXP, Renesas, AMD (Xilinx)…) for the automotive market.

From: SBP (Seirios)
Date: Tue, 2 Dec 2025 07:25:37 +0000

@JeremyGrosser Thank you for the info and the papers! Very interesting.

@F-Loyer Yes, that's what I wonder too. How does a software update relate to the potential impact of solar flares? A change in data redundancy? Error correction? Storage medium (less shielded memory chip)?

From: nbjessen
Date: Sun, 7 Dec 2025 15:18:30 +0000

Yeah, but you need /three/ CPUs. When one gets thrown off, you need to know which one is wrong. You need to be able to keep going w/o stopping.

Also: you need to make sure that you have ECC on all your memory, data-buses, etc. going in/out of the CPUs, because if there's a SEU on any shared data, both CPUs will be wrong (but also in sync).

I spent several years designing SEU-tolerant space HW. It's very easy for people to miss things.

What you really need to do is use a fault-injection tool to inject SEU all over, so that you can verify it gets detected and corrected. It's just not realistically possible to think of all the cases.

Ada and Other Languages

Development Costs and Other Benefits of Ada or SPARK vs. Other Languages

From: Kevin Chadwick (kevlar700)
Subject: Comparing the development costs and other benefits of Ada or SPARK vs other languages
Date: Thu, 6 Nov 2025 17:04:34 +0000
Newsgroups: forum.ada-lang.io

There is now a podcast format of the rational cost analysis site content that can be found here. Generated by notebookLM (AI) but useful when driving etc.. An interesting point I didn't pick up on when reading was that they threw in experienced C devs learning on the job with Ada and they were twice as effective as the same devs in a world class C environment.

Apple Podcasts - Comparing Development costs of C and Ada at Rational Software '95 [1]

Podcast Episode · Beyond memory safety with the Ada SPARK programming language

· 09/26/2025 · 52m

https://podcasts.apple.com/us/podcast/comparing-development-costs-of-c-and-ada-at/id1739672939?i=1000728630736

From: Luke (Lucretia)
Date: Sat, 8 Nov 2025 22:55:58 +0000

Needs updating for C++ and Rust.

From: nbjessen
Date: Fri, 5 Dec 2025 15:18:46 +0000

The data I've seen is that Rust is 10-30% more productive than C++. Tho that was from pro-Rust people. And I don't know how they collected the data: was it novice C++ vs. guru-level Rust programmers?

In any case: verification is 80% of pre-release product development. Property checking is 5 /times/ more productive (500%, not 50%) than writing tests. And Ada/SPARK is the only SW language that has it (VHDL and Verilog also do).

So we're talking an order-of-magnitude difference in productivity for Ada/SPARK.

For pre-release costs.

Now the cost of finding/fixing a bug post-release for SW is 10X higher than pre-release (rule-of-thumb) in lost sales, reputation damage, and possibly legal liability. If people get injured or killed, it can go as high as prison sentences in the US (criminal, not civil). Since property-checking achieves a dramatically higher level of quality, the risks drop proportionally.

From: John Perry (cantanima)
Date: Fri, 5 Dec 2025 17:53:06 +0000

> The data I've seen is that Rust is 10-30% more productive than C++. Tho that was from pro-Rust people. And I don't know how they collected the data: was it novice C++ vs. guru-level Rust programmers?

I don't have a scientific study for you, but judging from a Rust team I work with, of whom only 1 had worked with Rust earlier than 2020, and a C++ team we interface with, which contains at least some programmers that have been working with both C++ and that particular system for 10 years or more, 10-30% is a gross underestimate of how much more productive Rust programmers can be than C++.

That said there are other factors involved: our team has a strong Agile workflow and we frequently work in groups on our code. Their team is highly siloed and their "Agile" is not very. Our "genius"-level developers tend to be pretty loyal, rather than leave for greener pastures in a different part of the country. Their team is using QT, which is apparently incredibly incompatible with itself, to the point where version x.y.z is apparently incompatible with x.y.w.

So it's not a great comparison, but it's a comparison.

From: nbjessen
Date: Fri, 5 Dec 2025 19:38:29 +0000

An MIT study found that the range of productivity of individual SW engineers varied 50:1. The best did in a week what the worst did in a year. For teams, it was 2,000:1. In other words, culture >> technical ability by 40X. And that'll swamp any differences in tools, languages or design practices. Just FYI. As a self-made many-millionaire once told me: "The key to success is being able to first, get a team of brilliant people working for you, and two making sure they don't kill each other".

From: CUP (cup)
Date: Thu, 27 Nov 2025 11:20:22 +0000

Has re-training been factored into the development costs? The project needs to factor in the cost of training someone with no previous experience of Ada. In the UK, very few companies are willing to train people. It is normally pick it up as you go along, which is the wrong way of doing things since they will still be thinking in a different language but coding in Ada.

The other problem is recruitment. If you put down Ada, the recruitment companies will use that as a filter. That means suitable candidates will be excluded just because they don't have Ada. It may cost less to do a project in Ada but how easy is it to find the people. I've seen the same Ada advert (words are the same) from different recruitment agencies for one company for over a year.

From: Kevin Chadwick (kevlar700)
Date: Thu, 27 Nov 2025 13:14:41 +0000

> Has re-training been factored into the development costs?

It was. The two factor gain was with C devs learning Ada on the job. I do expect Rational had a certain number of experienced Ada devs that could help with that though. AdaCore provide training and mentors as an alternative. I had to teach myself and I'm sure that would have an impact but likely with other benefits. Nvidia boot strapped their own training with AdaCores services.

From: Jeff Carter (JC001)
Date: Fri, 28 Nov 2025 10:01:38 +0000

> It was. The two factor gain was with C devs learning Ada on the job.

You might also want to consider McCormick's real-time software [1] course. The U used C for all its lower-level courses, so all the students had C experience. None of them had Ada experience. No one ever finished the project using C, even when given 60% of the professor's solution, so we don't have

a factor, but many students were able to learn Ada and finish the project in Ada, even when given only a small percentage of the professor's solution, so clearly learning Ada and using Ada needs less effort than knowing C and using C.

[1] https://www.adaic.org/wp-content/uploads/2021/06/J-McCormick-XTalk.pdf

From: Kevin Chadwick (kevlar700)
Date: Fri, 28 Nov 2025 12:24:11 +0000

[...] I can't find the code he gave to his students unfortunately.

https://www.cs.uni.edu/~mccormic/RealTime/

From: Kevin Chadwick (kevlar700)
Date: Fri, 28 Nov 2025 12:34:26 +0000

This one made me laugh

The first 90% of the code accounts for the first 90% of the development time. The remaining 10% of the code accounts for the other 90% of the development time. (Tom Cargill)

Software Engineering Humor

https://www.cs.uni.edu/~mccormic/humor.html

From: Jeff Carter (JC001)
Date: Sat, 29 Nov 2025 10:15:29 +0000

The old 90-90 rule. I like this observation from Kernighan and Plauger (/Software Tools/):

> Most users of a tool are willing to meet you halfway; if you do ninety percent of the job, they will be ecstatic.

Put the two together and you only need 90% of the development time.

Rust Took Out Cloudflare

From: Luke (Lucretia)
Subject: Rust Took Out Cloudflare
Date: Wed, 19 Nov 2025 09:19:18 +0000
Newsgroups: forum.ada-lang.io

I don't know Rust, could this have happened with Ada?

Cloudflare outage on November 18, 2025 [1]

Cloudflare suffered a service outage on November 18, 2025. The outage was triggered by a bug in generation logic for a Bot Management feature file causing many Cloudflare services to be affected.

[1] https://blog.cloudflare.com/18-november-2025-outage/#memory-preallocation

From: Fernando Oleo Blanco (Irvise)
Date: Wed, 19 Nov 2025 09:40:10 +0000

I did not read much further than the headline that it was .unwrap(). AFAIK, yes, it could have easily happened to Ada. Basically, .unwrap() says "I know this could throw an exception, but I am not going to care/handle it". So the exception that was generated went up the program/system and crashed it. Basically, Ariane 5 version 2.0.

Could have Ada prevented it? No, bad code is bad code. Could have SPARK prevented this? Yes, with the Exception analysis system that was integrated a couple of years ago, one could have caught these issues.

From: Luke (Lucretia)
Date: Wed, 19 Nov 2025 09:59:52 +0000

Ariane blew up because the management decided to save money (look at how well that worked out) by utilising a package from the previous Ariane which was incompatible.

From: waleedmebane
Date: Wed, 19 Nov 2025 15:45:40 +0000

An unwrap doesn't cause an exception since Rust doesn't have exceptions. It causes a panic. Panics terminate a process, and generally cannot be handled. That's why I wouldn't use Rust in a server. In a language that raises/throws exceptions, there could be (handle-/catch-all or handle-/catch-most) exception handlers at boundaries that allow for restarting tasks. [...] But that might not be possible in general with Rust panics. Therefore, it doesn't take extraordinary discipline of handling all error conditions at the most specific/appropriate points in the code.

I have heard that the Linux kernel developers did not allow Rust into the kernel until a special version without panics was written because they were unwilling to allow the Rust panics into the kernel.

From: César Sagaert (csagaert)
Date: Wed, 19 Nov 2025 15:56:27 +0000

> [...] handlers at boundaries that allow for restarting tasks.

This is precisely what libraries like tokio do in Rust. Panics are uncatchable in very specific conditions:

- a panic in a destructor during unwinding cannot be caught (double panic, essentially)
- no panics can be caught when compiling with panic=abort (and you wouldn't compile a server with that)
- a panic across the FFI barrier cannot be handled properly

But panics are not a control flow feature, and when a function panics it usually means that the program is in an unrecoverable state. The use of unwrap() by cloudflare was bad error handling that no language can save you from.

From: John Perry (cantanima)
Date: Wed, 19 Nov 2025 18:27:11 +0000

> An unwrap doesn't cause an exception since Rust doesn't have exceptions. It causes a panic. Panics terminate a process, and generally cannot be handled. That's why I wouldn't use Rust in a server.

I work on a team that uses Rust to build a server, and it never panics from an .unwrap because our coding standard & peer review forbid the use of .unwrap, so that we therefore handle or propagate every single Result and Option type. Rust's type system /forces/ us to handle those objects: they're right there in the function signature. We've gotten pretty good at that; I can't remember the last time I had to PR a commit where a lazy dev tried to slip an .unwrap into the code.

What's more, the standard Rust toolchain pitches a fit if you don't document the possibility of an .unwrap or a .panic.

By contrast, Ada doesn't force you to handle exceptions, and its function signatures don't even indicate when a function might raise an exception, so I suppose it would be fair to say that I would much rather write a server in Rust than in Ada?

(I don't know if it's fair to say that the standard Ada toolchain, i.e., GNAT, won't pitch a fit if a function either raises an undocumented exception or fails to handle an exception raise by a function it calls, but I don't remember it ever complaining about it. Then again, neither would I seriously say that I would rather write a server in Rust than in Ada.)

The Rust-based server I work on /can/ panic on account of arithmetic under/overflow, index errors, etc., which is why we eliminate most indexing via the use of iterators. (If we have to index, we use .enumerate, so whose indices are guaranteed correct.)

Again, Ada will likewise terminate in such circumstances, and it has no .enumerate capability to my knowledge – there's 'First and 'Last for arrays and .First_Index and .Last_Index for Ada.Containers.Vectors, which is both inconsistent and a little harder to work with.

Added later: I concede that .enumerate will not solve all issues with indexing; I can readily remember a project I worked on where that would have been impossible. In our use case, however, it's borne the brunt of most of our problems, and .clamp has usually done the rest.

From: waleedmebane
Date: Wed, 19 Nov 2025 19:05:59 +0000

> Again, Ada will likewise terminate in such circumstances, and it has no .enumerate capability to my knowledge – there's 'First and 'Last for arrays and .First_Index and .Last_Index for Ada.Containers.Vectors, which is both inconsistent and a little harder to work with.

Unfortunately I don't know Rust very well even though I read the Rust book a long time ago. I'm also new to Ada. Is

.enumerate a way to do a foreach loop? Is for I in My_Array'Range loop not an Ada equivalent?

From: John Perry (cantanima)
Date: Wed, 19 Nov 2025 19:36:27 +0000

> Is .enumerate a way to do a foreach loop?

Not sure how to answer, but it works like this:

```
for (ith, element) in impls_into_iter {
  assert!(imlps_into_iter[ith] == element);
}
```

> Is for I in My_Array'Range loop not an Ada equivalent?

No.

- For an array, it's /close/, but you still have to declare a variable and reference the element.
- For a vector, or any other container, there is no such construct.

From: waleedmebane
Date: Wed, 19 Nov 2025 19:49:06 +0000

I think there is something like that in Ada 2012:
for E: Typename of Iterable **loop**

From: John Perry (cantanima)
Date: Wed, 19 Nov 2025 20:28:43 +0000

> for E: Typename of Iterable loop

How does this obtain E's index?

From: waleedmebane
Date: Wed, 19 Nov 2025 20:29:48 +0000

It doesn't. Sorry. I missed that. It only gives the element.

Ada Practice

Aspect Dimension_System with a Single Dimension

From: Markus A. Stokkenes (mstkns)
Subject: Aspect Dimenision_system with a single dimension
Date: Tue, 11 Nov 2025 14:38:25 +0000
Newsgroups: forum.ada-lang.io

I am playing around with GNAT's dimensionality checking for physical quantities, which are implemented with the aspects Dimension_System and Dimension, see https://blog.adacore.com/uploads/dc.pdf.

I would like a type which models geometric quantities, i.e. angle, length and volume. For that, we need only a length dimension. However, I can't get the following example to compile:

```
package Geometry is
  type Quantity is new Float with
    Dimension_System => (
      (Unit_Name => Meter,
       Unit_Symbol => 'm',
       Dim_Symbol => 'L')
  );
end Geometry;
```

Trying to compile this yields

geometry.ads:5:10: error: expected positional aggregate

Can anyone tell me what's going on here? I am using GCC GNAT 14.2.0 by the way.

From: Nordic_Dogsledding
Date: Tue, 11 Nov 2025 15:33:42 +0000

> geometry.ads:5:10: error: expected positional aggregate

You have to give the parameters in the correct order (contrary to other aggregates).

See a critical review of this method's achievements and shortcomings in my paper Physical units with GNAT, which has appeared in /Ada User Journal 35.1/).

http://archive.adaic.com/tools/CKWG/Dimension/Physical_units_with_GNAT_GPL_2013-AUJ35.1.pdf

As alternatives, you can try Dmitry Kazakov's Units of measurement for Ada [1] or GitHub - CKWG/SI_Units-Checked-and-Unchecked: Ada packages for handling SI units of measure [2].

[1] https://www.dmitry-kazakov.de/ada/units.htm

[2] https://github.com/CKWG/SI_Units-Checked-and-Unchecked

Artificial Intelligence for Ada Coding?

From: VMo
Subject: Do You Use Artificial Intelligence for Ada Coding?
Date: Tue, 9 Dec 2025 08:05:29 +0000
Newsgroups: forum.ada-lang.io

As I am still alone coding the TLALOC Ada 83 compiler, I tried to get help from Claude AI for coding in Ada 83. Does anyone use AI for Ada coding ? Ada 83 is especially suited to AI assisted programming with its simple package structure. You define the specification, revise it with AI and then let the AI code the body under feedback supervision. It works extremely well.

From: Quentin (Heziode)
Date: Tue, 9 Dec 2025 08:23:55 +0000

I personally tried this for FOSS or a personal project.

I do not use it for industrial professional projects. Maybe in the future, if we can have a fine-tuned LLM well trained on Ada that can run on our computer / our company server, not on a remote LLM from another company.

On my side, I tried the following LLM:

- ChatGPT 5.1 (high reasoning): Meh… it needs more iterations than Claude for tasks
- Gemini 3: good but not the best
- Claude Sonnet/Opus: I use it for coding since 3, now with 4.5 it's awesome

For your purpose, you have to give it a try because those LLM are trained on public code and ARM, so, it will try to use the latest language version possible (when it does not generate malformed code).

Furthermore, I'm exploring the uses of multi-agents systems and spec-driven-development.

When I 1st saw "spec-driven-development" I laughed a lot, because, this is what we've been doing with Ada… for decades

Currently, I am experimenting with the use of BMAD-METHOD [1] on Windsurf and Claude Code. It is too heavy for small projects but is ok for larger ones. IMHO, we have to add custom agents for Ada (and SPARK) coding, but the default does the job… well.

I also quickly tried spec-kit [2] and openspec [3] but I am unable to do efficient things with it (the goal of LLM is to save time, not waste it).

In any case, you will have to review the code it write because it can be too verbose, sub-optimal, or not doing what you want (rare with a spec-driven-framework).

[1] https://github.com/bmad-code-org/BMAD-METHOD

[2] https://github.com/github/spec-kit

[3] https://openspec.dev

From: Stéphane Rivière (NumberSix)
Date: Tue, 9 Dec 2025 08:55:07 +0000

Since search engines have become useless, I use AI to:

- Search the web
- Create summaries (give me an overview of this API with CRUD examples)
- Get debug snippets on environments I don't know (and don't want to know)

In short, to save a lot of time on acquiring/managing "stupid knowledge."

I also use it in "rubber duck" mode. I explain my problem to AI, and it comes up with a solution strategy. Sometimes it's silly (often depending on my own prompt), sometimes it gives me ideas

The only time I consulted her for Ada, she hallucinated in a very convincing way.

In short, AI has become indispensable, like fire.

And the same consequences in case of misuse.

From: John Perry (cantanima)
Date: Tue, 9 Dec 2025 15:57:10 +0000

I don't use it to code anything (yet). A lot of what I do relies on mathematics, and my impression, as well as my experience, is that AIs are still terrible with mathematics. (They don't have actual

reasoning skills, if by “reasoning” we mean logical deduction.) Besides, I enjoy the process of devising the solution /and/ writing the code; that’s one reason I do the Advent of Code problems.

Last year I worked with an entry-level programmer whose community college programming class apparently consisted in teaching students how to query an AI to write the program for them. (“apparently” is load-bearing; I have no idea whether that’s what actually happened.) It was pretty impressive what the AI could do. *But* when it provided him invalid code, he was incapable of fixing it. I sat with him once to debug it and it was a 2-hour-long nightmare.

I am interested in using AI for code quality issues, and if AI-generated code produces decent enough comments that the debugging experience would have been easier, my attitude would change a bit. But only a bit.

From: OneWingedShark
Date: Wed, 10 Dec 2025 00:08:06 +0000

> [...] consisted in teaching students how to query an AI to write the program for them.

This is how you churn out people with no competence.

From: John Perry (cantanima)
Date: Tue, 9 Dec 2025 16:01:28 +0000

> You define the specification, revise it with AI and then let the AI code the body under feedback supervision. It works extremely well.

To follow up: I sometimes do something like this with a different task at work, and I find more often that it’s much quicker to do it myself than put up with the give and take of prompt engineering. How do you feel about the efficiency of prompt engineering with AI 83 code?

(Some people say that AI’s will relieve us of the tedium of certain uninteresting tasks, but that’s only an improvement if the tedium of prompt engineering can be minimized.)

UTF-8, Identifiers and Dealing with Strings in Ada

From: Ada Orbit (AdaOrbit)
Subject: UTF-8, identifiers and dealing with strings in Ada
Date: Tue, 16 Dec 2025 11:47:40 +0000
Newsgroups: forum.ada-lang.io

I also had a lot of problems with string handling in Ada during AoC. I guess I am just not used to “the Ada way” of parsing and filtering.

I can only agree with what you wrote about embedded! Going over the SVD generated files it becomes clear how Ada's design was made to be used for embedded.

From: Dmitry Kazakov (dmitry-kazakov)
Date: Tue, 16 Dec 2025 12:12:42 +0000

> I also had a lot of problems with string handling in Ada during AoC. I guess I am just not used to “the Ada way” of parsing and filtering.

There is no “Ada way of parsing.” There are unexplainable ways people tend to do parsing in the most artificial and complicated ways. It must absolutely include multiple stages of meaningless actions, producing intermediate results of no use and of course never done in a direct imperative way: get a thing, advance, repeat. No, it must be some intricate combination of functions, scanners, filters, tokenizers with parameters tuned in some magical way to produce results. When Ada offers a straightforward elementary approach people simply fail to grasp the idea. That must be “the Ada way!” Though the same approach worked perfectly in C before Ada. In C that does not have strings! Parsing without handling strings…

It is not the Ada way, it is just a sane way.

From: Michał (pmnw)
Date: Tue, 16 Dec 2025 14:07:03 +0000

Ada/SPARK, marketing, reach and the abysmal state of things [1]

> I have tried to do the advent of code with Ada also but because the challenges are mostly string handling I find it very annoying to do.

> I also had a lot of problems with string handling in Ada during AoC.

Dealing with strings in Ada is simple as long as one remembers that strings are arrays and that Ada supports array slicing.

[1] https://forum.ada-lang.io/t/ada-spark-marketing-reach-and-the-abysmal-state-of-things/4019/18

From: Dmitry Kazakov (dmitry-kazakov)
Date: Wed, 17 Dec 2025 19:58:23 +0000

The point about slicing was important. That is one of the reasons why unbounded strings are not needed. You never ever copy anything when parsing. You can always pass a source slice to a subprogram. Ada slicing keeps index invariant, which additionally simplifies parsing because you do not need to shift indices back and forth; you can pass in and out them together with the slices as is.

From: Kohlrak
Date: Thu, 18 Dec 2025 06:39:53 +0000

Speaking of strings, I’m told Ada can’t handle utf8 without first holding it as wide_wide_string and converting to one of the other string types. I imagine that would’ve been simple, and i’m surprised that since i could use something like 猫 as an identifier in the language itself that string literals aren’t so simple. If I have a program with a lot of instructions for the user that can add up over time.

From: Dmitry Kazakov (dmitry-kazakov)
Date: Thu, 18 Dec 2025 09:02:36 +0000

> Speaking of strings, I’m told Ada can’t handle utf8 without first holding it as wide_wide_string and converting to one of the other string types.

You were lied to. On the contrary, you should never use Wide_Wide_String, forget it exists. Historically Ada predates Unicode. When Unicode came, a quick and dirty decision was made and Wide_String (UCS-2) was introduced. BTW, Microsoft made the same error in Windows. Then there was no way back and so Wide_Wide_String (UCS-4) came. Microsoft cared less and instead of keeping it compatible simply declared, oops, it is UTF-16 not USC-2. Why they simply did not go all the way back to UTF-8 is beyond me. But we *can*! Now simply ignore all that mess and consider Ada String UTF-8.

All text processing algorithms work in UTF-8 without changes. For conversions to legacy code pages, code points, sets, maps, normalization see: String processing for Ada [1] .

Keep your program ASCII-7. UTF-8 literals are no problem with above, however it is not a good idea if you mean localization.

[1] https://www.dmitry-kazakov.de/ada/strings_edit.htm

Ada Frontiers

Ada 2022 'parallel' Implementation Beta for FSF GCC/GNAT

From: Dirk Craeynest <dirk@orka.cs.kuleuven.be>
Subject: Ada 2022 'parallel' implementation beta for FSF GCC/GNAT
Date: Tue, 16 Dec 2025 09:24:29 +0000
Newsgroups: comp.lang.ada

Great news for the Ada community!

Many Ada users were excited about the support for light-weight parallelism in the Ada 2022 standard, but disappointed it was not yet implemented in an Ada compiler.

ARG member Richard Wai recently posted announcements about support for the Ada 2022 parallel "for" and parallel "do" features. A copy is provided below for your information.

Richard wrote: "We would love to have more members of the Ada community try-out these features." If you do, please keep us all informed!

Dirk Craeynest

[Copied from the ada-lang.io forum [1] -- dc]

Ada 2022 'parallel' implementation beta for FSF GCC/GNAT

Richard-Wai

I am very pleased to announce that the core Ada 2022 "parallel" features have been implemented for mainline FSF GNAT as part of a successful Google Summer of Code project. The patch is now ready for beta testing.

We are preparing to formally submit this patch to the FSF GCC project, to have it incorporated into GCC trunk, and therefore all future FSF GCC releases. Before we make that submission, we hope to seek additional feedback from the Ada community.

This patch introduces most core capabilities for the parallel keyword, including:

* Parallel loops

* Parallel blocks

* Early exit

* Chunking

This patch does NOT yet support Parallel Iterators - but this is being actively worked on and hopefully will be completed soon.

The GSoC project work was hosted on the Ada Rapporteur Group's own GCC mirror GitHub repo, and the stable version of the parallel beta release currently lives at [2].

This branch can be built and bootstrapped as-is on most mainstream platforms using FSF GCC 15, with a standard build process. No additional libraries or build flags are needed, as the parallel features do not imply any new compiler, runtime, or platform dependencies.

Additionally, Maxim Reznik has made available binary builds of the parallel support beta for most popular platforms. He also includes instructions on how to get going with Alire. Look for the "GCC with parallel PREVIEW" release at [3]. Expand the "Assets" area at the bottom of the section to download binary builds for your platform.

By default, GNAT will expand parallel loops/blocks into sequential (regular) loops resp. blocks. To get actual parallelization of parallel constructs, the existence of an Ada "light weight threading" library (formally an Ada subsystem a la the Ada Reference Manual 10.1-3) is required. GNAT will detect the presence of the "LWT" subsystem at compile-time, and if "withed", will generate calls to the LWT subsystem during expansion. It is therefore conventional that the unit containing the main subprogram "withs" LWT. Refer to the example programs included with the reference LWT subsystem to see how this works.

A reference LWT subsystem implementation currently lives under the Parasail language project [4]. This may be separated from the Parasail repository at a later time. This LWT implementation is also an official Alire crate of the same name, and Maxim's instructions detail how to install LWT and use the beta toolchain under Alire.

The reference LWT subsystem could potentially be incorporated into the GNAT standard library (libgnat) at some point in the future, but it was decided to keep the first iteration as simple and digestible as possible.

This is only the first phase, and we look forward to additional refinements in the future!

[1] https://forum.ada-lang.io/t/ada-2022-parallel-implementation-beta-for-fsf-gcc-gnat/4054

[2] https://github.com/Ada-Rapporteur-Group/gcc-mirror/tree/devel/arg-proto/ada2022-parallel-release

[3] https://github.com/reznikmm/GNAT-FSF-builds/releases

[4] https://github.com/parasail-lang/parasail

Note that Alire is NOT required to beta test this build, and simply making the sources of the LWT reference implementation available to the compiler is sufficient (using -I for GCC or gnatmake).

There are also some Ada 2022 parallel example programs under lwt/a22_examples, and these can be build and run with the vanilla FSF GNAT toolchain as follows (Linux/UNIX):

```
$ git clone https://github.com/parasail-lang/parasail
$ cd parasail/lwt/a22_examples
$ gnatmake -gnat2022 -I../ n_queens.adb
$ ./n_queens
```

We would love to have more members of the Ada community try-out these features. Please let me know if you need any support getting set up, or if you have any other questions.

SPARK/Ada

Validate Float Type Implementation

From: nerd_type
Subject: SPARK: Validate float type implementation
Date: Fri, 28 Nov 2025 16:47:54 +0000
Newsgroups: forum.ada-lang.io

I would like to use SPARK to prove that the underlying implementation uses IEEE754 for the Float type.

The approach for validation is to compare the data used by the float type with a reference byte array representation [1].

The trouble is that SPARK will not allow the address overlay for the byte array / float type.

[Code omitted here. —arm]

[1] IEEE-754 Floating Point Converter https://www.h-schmidt.net/FloatConverter/IEEE754.html

From: Dmitry Kazakov (dmitry-kazakov)
Date: Fri, 28 Nov 2025 17:15:59 +0000

The standard does not specify endianness, so in general you cannot do it the way you described because there is no guarantee that machine endianness is the same.

As I side note, I have no idea why you want this, because in communication and automation we rather immediately convert IEEE 754 type to the problem space Ada type and back.

For conversions of IEEE types see: Simple components for Ada [1].

Also we never use IEEE 754 without killing non-numbers first. Like this:

```
type Safe_Float is new Float range Float'First..Float'Last
```

I have horror stories to tell about NaN slipping into the database ruining a week of mileage tests (one hour of the dynamometer is about 10K EUR).

Returning to the question, the best method was to check the range since different floating-point formats have different ranges or else to check for non-numbers. Only IEEE 754 has them AFAIK.

[1] https://www.dmitry-kazakov.de/ada/components.htm#IEEE_754

From: Kevin Chadwick (kevlar700)
Date: Fri, 28 Nov 2025 17:40:27 +0000

> The standard does not specify endianness

There is Scalar_Storage_Order though which can perform byte swapping for you as needed on the array but I don’t know if it works with overlays.

Lady Ada Mediates Peace Treaty in Endianness War | AdaCore [1]

[1] https://www.adacore.com/papers/lady-ada-mediates-peace-treaty-in-endianness-war

From: Dmitry Kazakov (dmitry-kazakov)
Date: Fri, 28 Nov 2025 20:02:30 +0000

What about Shift_Left and *or*?

BTW, who said it is big or little endian? You know, some I/O terminals have bytes of IEEE 754 single precision floats arranged in the most peculiar order.

From: damaki
Date: Fri, 28 Nov 2025 18:31:10 +0000

That warning is happening because SPARK does not permit Float values to have invalid bit patterns such as NaN. Using an address clause to create a mutable alias could allow Sample to be

set to an invalid bit pattern by writing certain values to Sample_u8.

If all you want to do is read the bit pattern (not write), then you can make Sample and Sample_u8 constant to prevent this warning.

From: damaki
Date: Sat, 29 Nov 2025 10:52:17 +0000

To avoid endianness issues you could use an Unsigned_32 instead of a byte array. IIUC Unsigned_32 and Float should have the same byte order on the same machine in practice.

From: Dmitry Kazakov (dmitry-kazakov)
Date: Sat, 29 Nov 2025 10:59:54 +0000

I think a good portable test could be:

```
Float'Succ (0.0) = 2#1.0#E-149
```

it covers three key components the radix (2), exponent (8-bit) and mantissa (24), which should give different results for:

- PDP-11 was a precursor of IEEE 754 but it had a different normalization which should result in a different value of Float'Succ (0.0).
- VAX was like PDP-11, I believe.
- IBM 360 had 24-bit mantissa and radix 16.
- PDP-8 had three 12-bit words and 12-bit exponent.

P.S. I am too lazy to verify this, e.g. under a VAX emulator. Anyway the task does not make sense to me.

From: nerd_type
Date: Mon, 1 Dec 2025 05:46:03 +0000

I simplified the operation to reduce the complexity of the proof. It is interesting to note that the Unchecked_Conversion version is prove-able and the version that uses a memory overlay cannot be proved.

[Code omitted here. —arm]

Is SPARK in Ada a Theorem Prover?

From: Srayan Jana (ValorZard)
Subject: Is SPARK in Ada a theorem prover?
Date: Fri, 28 Nov 2025 21:01:38 +0000
Newsgroups: forum.ada-lang.io

This might be a dumb question, but is SPARK in Ada similar to Z3 or lean 4? As in, can it be used to formally prove stuff like theorems or whatever?

From: Quentin (Heziode)
Date: Fri, 28 Nov 2025 21:13:38 +0000

Someone with more expérience in SPARK will give you an answer.

But, gnatprove that prove your SPARK code, uses z3, in addition with other tools:

https://docs.adacore.com/spark2014-docs/html/ug/en/source/how_to_run_gnatprove.html#running-gnatprove-from-the-command-line

From: damaki
Date: Fri, 28 Nov 2025 21:50:23 +0000

SPARK isn't a theorem prover, rather it's a programming language derived from Ada plus static analysis tools that allows you to prove things about programs such as absence of run-time errors and functional correctness. So you write SPARK code which can be compiled with a compiler such as GNAT, then use the GNATprove tool to perform static analysis on the code which will try to automatically prove various properties on the code.

Behind the scenes, GNATprove uses Why3 and automatic theorem provers like Z3, CVC5, Alt-Ergo, and COLIBRI to prove various checks on the code. For example, given the following function in SPARK:

```
function Increment (X : Integer)
return Integer is (X + 1) with Pre => X
   Increment'Result = X + 1;
```

GNATprove will emit a verification condition (VC) for the addition X + 1 to try to prove that the result of the addition fits within the bounds of type Integer, assuming that X from the precondition. Another VC will also be emitted for the postcondition to try and prove that the returned value is equal to X + 1.

So SPARK isn't used to prove mathematical theorems, but rather proves various properties about the program such as absence of run-time errors (no arithmetic overflows, buffer overflows, division by zero, etc) as well as functional properties expressed in contracts (preconditions, postconditions, invariants, predicates, etc).

That's a brief overview which I hope answers your question. If you want to learn more I recommend checking out AdaCore's Learn website. This page provides a nice overview of SPARK:

https://learn.adacore.com/courses/intro-to-spark/chapters/01_Overview.html

From: Srayan Jana (ValorZard)
Date: Fri, 28 Nov 2025 21:51:20 +0000

Hm, then I guess a follow-up question is, is there a way to do theorem proofs in Ada? Like a library for it or something?

From: A.J. Ianozi (AJ-Ianozi)
Date: Sat, 29 Nov 2025 01:29:51 +0000

This isn't specifically what you're looking for, but you can prove algorithms with SPARK:

AdaCore - I can't believe that I can prove that it can sort [1]

When an enthusiastic Ada programmer and a SPARK expert pair up to prove the most "stupid" sorting algorithm, lessons are learned! Join us in this…

I'm also aware of some crypto with his stuff written in SPARK, which result in them being provable (it's no longer maintained though)

GitHub - Componolit/libsparkcrypto: A cryptographic library in SPARK 2014 [2]

[1] https://www.adacore.com/blog/i-cant-believe-that-i-can-prove-that-it-can-sort

[2] https://github.com/Componolit/libsparkcrypto

From: Srayan Jana (ValorZard)
Date: Sat, 29 Nov 2025 06:51:38 +0000

This article is pretty close to what I was originally imagining SPARK to be:

AdaCore - Solving Sudoku with AdaSAT [1]

Seems like AdaCore already has a library that can do proof solver stuff?

[1] https://www.adacore.com/blog/solving-sudoku-with-adasat

From: Fernando Oleo Blanco (Irvise)
Date: Sat, 29 Nov 2025 07:45:35 +0000

AdaSAT is open and used in production [1].

There are libraries that already have structures fully proven that you can build on them and create more complex proofs, the main one being SPARKlib [2].

SPARK is more about proving code than rather proving the problem behind it. So in a way, SPARK is not like Rocq/Coq or Lean4, but it uses the same underlying principles and tools. If you want Satisfiability Theory (SAT) those systems may be better (you can check out SMT-Lib too) SMT-LIB The Satisfiability Modulo Theories Library [3] or Operational Research tools such as HiGHS.

EDIT: I incorrectly stated that AdaSAT was a Proof-of-Concept when in reality it is used in production as corrected by Fabien.

[1] https://github.com/AdaCore/adasat

[2] https://github.com/AdaCore/SPARKlib

[3] https://smt-lib.org/index.shtml

From: Srayan Jana (ValorZard)
Date: Sat, 29 Nov 2025 21:44:12 +0000

For context, what I'm trying to do is create a "proof solver" for Tetris.

Think something like stackrabbit for chess, but for Tetris instead.

So, given a Tetris board, is it solvable, or has it reached a fail state?

So, I would like to be able to "prove" that certain states are valid and other states aren't. Tbh, this feels like something more suited to Haskell, but I want to have a graphical display/simulation written in Raylib or SDL, and Haskell isn't great for actual gameplay code from what I've seen.

From: damaki
Date: Sun, 30 Nov 2025 09:41:01 +0000

That sounds doable in SPARK, and you might find this blog post interesting which implements Tetris in SPARK on an Arm Cortex-M4:

https://www.adacore.com/blog/tetris-in-spark-on-arm-cortex-m4

From: Fabien (Fabien.C)
Date: Mon, 1 Dec 2025 09:16:17 +0000

> AdaSAT is open but it was just a proof of concept

AdaSAT is used in production by langkit_support and all the libraries and tools that depend on it (Libadalang, VS Code plugin, GNAT Studio, GNATformat, GNATdoc, etc.).

See here: Solving Sudoku with AdaSAT | AdaCore [2]

[1] https://github.com/AdaCore/adasat

[2] https://www.adacore.com/blog/solving-sudoku-with-adasat

Ada/SPARK, Marketing, Reach and the Abysmal State of Things

From: Fernando Oleo Blanco (Irvise)
Subject: Ada/SPARK, marketing, reach and the abysmal state of things
Date: Sun, 7 Dec 2025 09:19:59 +0000
Newsgroups: forum.ada-lang.io

I am watching some of the new videos uploaded to YT in the ACM SIGPLAN channel [1] and it is just depressing. There are a ton of videos about formal proofs, better software quality, software analysis, etc; and SPARK is nowhere to be found AT ALL. Even in language or syntax comparisons and analysis. I blame the people presenting for not properly doing their research (only one video mentioned Ada [2], but in the context of the 80s). But I also blame us for doing the poorest job at spreading and sharing Ada/SPARK. And, no, AdaCore is not the one responsible for marketing, this is the responsibility of the community…

Here are some other videos that do not mention SPARK and if they would, their conclusions would have been completely different:

- https://www.youtube.com/watch?v=7i8P8HM4CLU
- https://www.youtube.com/watch?v=GptVarCy2v4
- And more, but these were the videos whose titles I was drawn to. You can find many more examples from previous years.

We really need people who are passionate and knowledgeable going into non-Ada circles and environments and showcasing and explaining Ada/SPARK in an engaging manner and in their full glory.

Also, to be fair to us, some "researchers" purposefully "miss/hide" some information to make their point be correct, as otherwise, their thesis or ideas would not be valid. I do not consider this to be a reasonable nor good thing, but it is what is happening in many areas. Most people who do that with Ada probably would use excuses such as "Ada is dead" or "nobody uses SPARK". We need to showcase the modern and updated uses of Ada/SPARK to shut up people about it (see all my other posts here about NVIDIA).

Anyway, I am quite frustrated and I would like to blame myself. The world may not be a nice, honest and fair place, but that is just the way things are. I want to focus on the things that I can actually change and have an impact on.

[1] https://www.youtube.com/@acmsigplan

[2] https://www.youtube.com/watch?v=E9oI_oVnFmY

From: Sebastián Benítez (sbenitezb)
Date: Sun, 7 Dec 2025 09:49:42 +0000

I would like to see some Linux services rewritten in SPARK, perhaps a distro where the core tools and services are rewritten in SPARK. If I had time to invest in this, I would start the project.

From: Fernando Oleo Blanco (Irvise)
Date: Sun, 7 Dec 2025 10:00:44 +0000

That is what the Rust people have been doing for years, they have rewritten sudo, coreutils and many many other common tools such as less, cp, grep, etc.

One of the biggest issues with Ada/SPARK is that it lacks libraries that can hugely speed up the development of such tools. Also, less people know it, there is less momentum, the Ada tooling is quite unique and people do not want to learn it and packaging Ada software is a huuuuge pain in the ass, so software written in Ada is a pain to get into the repos for people to use it without friction…

Though I also share your vision and goal, this was not criticism towards it.

EDIT: actually, the issues I explain for Ada also existed in other languages, such as Fortran. But that community got the memo, bonded over it and created a somewhat "extended standard library for fortran" and now everybody uses it and the community has joined into creating a "single source of truth" for those libs and their usage has become pretty common:

https://github.com/fortran-lang/stdlib

From: Sebastián Benítez (sbenitezb)
Date: Sun, 7 Dec 2025 10:07:22 +0000

I think rewriting ls like tools is useless. Sudo and services that might be exploitable are better targets. Also package manager, init system/service manager, libraries of course. The basics to get noticed and to get more people contributing.

From: Quentin (Heziode)
Date: Sun, 7 Dec 2025 10:25:22 +0000

> One of the biggest issues with Ada/SPARK is that it lacks libraries that can hugely speed up the development of such tools

Yes… that's very sad.

BUT, I am working on this alone, and working in my free time on this, notably with Adarium [1], which I hope to be able to make a living from one day.

Currently, I am working on Aclida, a formally proven, full-featured CLI framework. The sub crate aclida_cli will be comparable to clap-rs [2] or Cobra [3]. Aclida will be a "meta-framework" to handle many things around the CLI (like interactivity, component "UI", deep color management (more powerful than ansiada), etc), and the global meta framework, aclida, will be more like Rich [4] or Spectre.Console [5].

It is not yet released, nor open-sourced yet. Currently I am only working on aclida_cli. Maybe I should "reserve" the word on Alire

The full-feature CLI framework will be the building block of awesome apps. Like you said, Rust redev popular Linux commands and services in Rust. I also want to do that in Ada/SPARK. It is feasible, but, we need to dev the building blocks before going deeper…

[1] https://adarium.dev

[2] https://github.com/clap-rs/clap

[3] https://cobra.dev

[4] https://github.com/textualize/rich

[5] https://spectreconsole.net/

From: Gautier de Montmollin (zertovitch)
Date: Mon, 8 Dec 2025 19:22:41 +0000

[...]

I notice that there are currently 0 comments on each of those 3 videos. It would be an opportunity to drop some words. Most probably it won't reach or even less influence the professors those researchers work for, but why not give it a try?

Making a cool game or a useful application in Ada will have much more impact IMHO.

From: Fabien (Fabien.C)
Date: Tue, 9 Dec 2025 17:18:00 +0000

I mean, I am biased but I do think we have a cool project to promote Ada: weenoisemakers.com/pgb-1

https://weenoisemakers.com/pgb-1/

From: geewiz
Date: Tue, 9 Dec 2025 17:38:42 +0000

> I do think we have a cool project to promote Ada

Instantly wishlist-ed!

From: Rene Hartmann (rehartmann)
Date: Fri, 12 Dec 2025 20:14:49 +0000

There are many programming languages, and very few of them enjoy true popularity. The others are stuck in niches. Sometimes a language suddenly gets some attention among software developers because some popular application is written in it, or because it supports a programming paradigm which became popular and after some time it returns into near-to-obscurity. The fact that Ada is not popular is unfortunate but there are many, many languages sharing this fate, and it does not mean that something went terribly wrong.

Surely there are some factors working against Ada: To many its wordy, Pascal-like syntax just doesn't feel cool. Many people are OK with writing an application that just sort-of works and fixing the bugs (or rewriting the whole crap) later. Ada appeals to the opposite mindset. That its implementations lack automatic memory management means it will probably not be able to compete in areas dominated by languages like Java and C#.

That said, I think there are areas where Ada can show its strengths. That it is possible to prove the correctness of a program instead of writing tests for a lot of cases, hoping that you covered all important cases is something I would call a cool feature. But it is hard to do in practice, so formal verification may be restricted to areas where correctness is extremely important, like when human lives depend on it.

I think there are two things Ada programmers can do: Writing Ada versions of popular applications or libraries to demonstrate that it's no less useful than other languages, or trying to write some 'killer application'. The first is arguably easier to do. Choosing between the first and the second is up to each individual programmer.

From: Henrik (Finnland)
Date: Mon, 15 Dec 2025 22:26:29 +0000

I'm no expert either but I find Ada extremely useful in embedded. I come from a C background and it is a joy to be able to map everything over registers instead of having to do pointers and shifting. Makes things so much clearer. And mapping of protocols is so good. You describe the protocol packages in a record and overlay them on the incoming data. Everything is clear and understandable in plain english. Just my 2 c…

:edited to add:

I have tried to do the advent of code with Ada also but because the challenges are mostly string handling I find it very annoying to do. Mostly a skill issue but Python would be a much more sane language to solve such problems.